Titre: Title:	Exploitation d'une structure monotone en recherche directe pour l'optimisation de boîtes grises
Auteur: Author:	Catherine Poissant
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Poissant, C. (2018). Exploitation d'une structure monotone en recherche directe pour l'optimisation de boîtes grises [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3006/

UNIVERSITÉ DE MONTRÉAL

EXPLOITATION D'UNE STRUCTURE MONOTONE EN RECHERCHE DIRECTE POUR L'OPTIMISATION DE BOÎTES GRISES

CATHERINE POISSANT
DÉPARTEMENT DE DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
FÉVRIER 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

EXPLOITATION D'UNE STRUCTURE MONOTONE EN RECHERCHE DIRECTE POUR L'OPTIMISATION DE BOÎTES GRISES

présenté par : POISSANT Catherine
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. LE DIGABEL Sébastien, Ph.D., président
M. AUDET Charles, Ph.D., membre et directeur de recherche
M. JALBERT Jonathan, Ph.D., membre

REMERCIEMENTS

Merci à mon directeur de recherche Charles Audet pour son excellent support et pour la liberté qu'il m'a laissée tout au long de la réalisation de ce projet. Pour son travail, je lui décerne la note de A^+. Oups... Désolée, on dit plutôt A^* à Polytechnique ! Mille fois merci à Pascal Côté et son équipe chez Rio Tinto pour l'idée du projet et pour l'accueil très chaleureux au Saguenay. Merci aux membres du jury, Sébastien Le Digabel et Jonathan Jalbert pour la révision rigoureuse de ce mémoire. Merci à Christophe Tribes et Viviane Rochon Montplaisir de l'équipe NOMAD pour leur grande patiente face à toutes mes questions sur le logiciel.

Un merci particulier à mes amis du bureau Loïc, Émilie, Kenjy, Marie-Ange, Mathieu et Damoon pour m'avoir changé les idées et gardé ma santé mentale dans le vert.

Merci à ma famille et mon copain pour avoir su écouter.

RÉSUMÉ

Ce projet se situe dans un contexte d'optimisation de boîtes noires industrielles. Celles-ci peuvent contenir des simulations numériques et des effets stochastiques, ce qui rend leur évaluation en un point couteuse en temps et leur comportement peut être bruité. En particulier, nous nous intéressons à des problèmes où, à l'augmentation d'une variable, l'utilisateur est en mesure de prédire l'augmentation ou la diminution de la valeur de la fonction objectif ou de l'une des contraintes qui constituent la boîte noire. C'est le cas pour le problème `Kemano` fourni par les ingénieurs de l'aluminerie Rio Tinto qui les guide hebdomadairement dans leur prise de décisions sur la gestion de leurs systèmes hydroélectriques. Le sens physique donné aux variables et aux contraintes permet aux ingénieurs de se prononcer sur l'existence d'effets monotones dans leur programme. Nous référerons à ce type de problèmes sous le terme «boîtes grises». Selon les ingénieurs de Rio Tinto, une meilleure calibration du modèle `Kemano` par un algorithme d'optimisation tel que MADS, un algorithme d'optimisation par recherche directe, permet un gain annuel d'environ 150 000 mégawatts par rapport à une calibration manuelle. D'où l'intérêt de ce projet, dont l'objectif principal est d'aider MADS à trouver une solution réalisable le plus rapidement possible à l'aide de ces informations concernant l'effet de croissance des variables sur les fonctions de la boîte noire. Dans un contexte de boîte noire, il est sous entendu que «plus rapide» est synonyme de «moins d'évaluations de la boîte noire».

À cette fin, nous avons bâti une base théorique solide grâce à une étude approfondie de la monotonie sur des cônes de fonctions à plusieurs variables. De cette étude, découlent une matrice de tendance ainsi qu'une direction de tendance qui serviront à guider MADS lors de l'optimisation d'une boîte grise. Nous proposons deux algorithmes (LS et T) pour exploiter concrètement les informations sur la monotonie par MADS par accélérer sa recherche d'un minimum local réalisable. Afin de mettre à l'épreuve ces techniques, elles ont été programmées à des fins expérimentales dans la version 3.7.3 du logiciel NOMAD, une implémentation en `C++` de l'algorithme MADS. Trois problèmes tests, dont `Kemano`, ont servi à évaluer les performances de LS et T. Ces problèmes se sont avérés intéressants dans le cadre de ce projet, car les informations sur la monotonie ont été extraites de façons différentes : analytiquement, par échantillonnage et avec l'intuition de l'utilisateur. Selon la nature du problème, différentes conclusions ont été tirées sur les performances de LS et T.

ABSTRACT

This project is in the context of industrial blackboxes optimization. These may contain numerical simulations and stochastic effects, making their evaluation time-consuming and their behavior noisy. In particular, we are interested in problems where, when increasing a variable, the user is able to predict the increase or decrease in the value of the objective function or one of the constraints that constitute the blackbox. This is the case for the Kemano problem provided by engineers at the Rio Tinto aluminum smelter who guides them in their weekly decision-making on the management of hydroelectric dams. The physical meaning given to variables and constraints allows engineers to predict the existence of monotonic effects in their program. We will refer to this type of problems under the term "gray boxes". According to Rio Tinto engineers, a better calibration of the Kemano model by an optimization algorithm such as MADS, a direct search optimization algorithm, allows an annual gain of about 150,000 megawatts compared to a manual calibration. Hence the interest of this project, whose main objective is to help MADS find a workable solution faster with the information about the effect of a variable growth on functions of the blackbox. In a blackbox context, it is understood that "faster" is synonymous with "fewer blackbox evaluations".

The main objective of this project is to help MADS, a direct search optimization algorithm, to find a feasible solution as quickly as possible with this information. In a blackbox context, it is implied that "faster" is synonymous with "fewer evaluations of the function".

With this goal in mind, we have built a solid theoretical foundation through a thorough study of monotony on cones of multivariate functions. From this study, we derived a trend matrix and a trend direction that will guide MADS when optimizing a gray box problem. We propose two algorithms (LS and T) to concretely exploit monotonic information in MADS. In order to test these methods, they were programmed, for experimental purposes, in the version 3.7.3 of the NOMAD software, a C ++ implementation of the MADS algorithm. Three test problems, including Kemano, were used to evaluate the performance of LS and T. These problems have been interesting in the context of this project because the information on monotony has been extracted by different ways: analytically, by sampling, and from the intuition of the user. Depending on the nature of the problem, different conclusions have been drawn about the performance of LS and T.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

DFO	Optimisation sans dérivées (*Derivative Free Optimization*)
CS	Recherche par coordonnées (*Coordinate Search*)
GPS	Recherche par motifs généralisée (*Generalized Pattern Search*)
PB	Barrière progressive (*Progressive Barrier*)
MADS	Recherche directe sur treillis adaptatif (*Mesh Adaptive Direct Search*)
NOMAD	Programme d'optimisation non-linéaire utilisant l'algorithme MADS (*Nonlinear Optimization by Mesh Adaptative Direct Search*)
T	Ordonnancement de S^k par $-d_T$
LS	Recherche linéaire dans la direction $-d_T$ (*Line Search*)
$f(x)$	Fonction définissant le problème d'optimisation
$f_0(x)$	Fonction objectif
$f_j(x)$	$j^{\text{ième}}$ contrainte relaxable et quantifiable
l_b	Borne inférieure sur les variables
u_b	Borne supérieure sur les variables
J	Ensemble d'indices
Δ^k	Taille du trellis à l'itération k
δ^k	Taille du maillage à l'itération k
D^k	Ensemble des directions de sonde à l'itération k
M^k	Ensemble des points du trellis à l'itération k
P^k	Ensemble des points de sonde à l'itération k
S^k	Ensemble des points de la recherche globale à l'itération k
Ω	Ensemble des points réalisables
X	Sous-ensemble de $\mathbb{R}^n$ comprenant les points respectant les contraintes non relaxables et non quantifiables
x^0	Point de départ
x^k	Meilleur solution à l'itération k
m	Nombre de contraintes relaxables et quantifiables
n	Nombre de variables
$R_X(x)$	Ensemble des directions réalisables à un ensemble X en un point x.
$\partial\Omega$	Frontière du domaine Ω
K	Cône convexe
$\text{conv}(X)$	Enveloppe convexe
T	Matrice de tendance

T_j	$j^{\text{ième}}$ vecteur colonne de T
K_{T_j}	Cône de tendance construit à partir de T_j
d_T	Direction de tendance

CHAPITRE 1 INTRODUCTION

L'optimisation de boîtes noires est un sujet de recherche qui croît fortement en popularité depuis les années 90 avec la sophistication des ordinateurs. Les ingénieurs sont maintenant en mesure de créer des programmes plus performants capables de simuler des phénomènes physiques complexes. En résulte ainsi un désir accru d'en optimiser les paramètres [22, 50]. Cependant, les irrégularités issues de simulations physiques, ainsi que la complexité des algorithmes, rendent bien souvent les méthodes d'optimisation classiques, telles que les multiplicateurs de Lagrange [36], inapplicables. Plus particulièrement, le gradient de la fonction peut être peu fiable, trop coûteux en temps à calculer ou même inexistant, d'où l'intérêt grandissant envers l'optimisation sans dérivées (DFO). Quelques familles d'algorithmes se trouvent dans cette catégorie. Parmi les plus connues, on y retrouve : les algorithmes génétiques [37], la méthode de Nelder-Mead [49], les régions de confiance [24] et les algorithmes par recherche directe [39]. Ce projet est uniquement concerné par cette dernière famille en raison de l'existence de preuves de convergence théoriques ainsi que par la flexibilité permise dans leurs implémentations et l'ajout d'extensions [8, 57].

Le terme boîte noire réfère à des problèmes dont la forme analytique est jugée tellement complexe ou inexistante qu'elle devient inexploitable. Il est donc souvent pris pour acquis qu'aucune hypothèse sur le comportement du problème ne peut être posée. Cependant, le sens physique donné aux variables permet parfois à l'utilisateur de la boîte noire de prévoir leur impact sur l'accroissement des contraintes ou de la fonction objectif. Un exemple concret serait celui du respect du budget alloué à un chantier de construction en fonction de la largeur de la route à construire. Il est évident qu'une route plus large sera plus dispendieuse et ce, indépendamment de tout autre paramètre fixé. Il existe alors une structure monotone sous-jacente à cette boîte noire lui conférant ainsi le statut de boîte grise puisque certaines informations sont maintenant connues. Il est à noter que notre définition de «boîte grise» diffère de ce qui est généralement retrouvé dans la littérature [16,17,19]. Bien qu'un consensus semble tenir pour le terme «boîte noire», les auteurs d'articles doivent préciser ce qu'ils entendent par boîte grise. Par ailleurs, une discussion complète sur le sujet est présentée dans la section 2.2. Pour l'instant nous nous contenterons de préciser que, dans ce travail, le terme «boîte grise» réfère à des problèmes de type boîtes noires dotés d'une structure monotone complète ou partielle connue par l'utilisateur. C'est-à-dire que toutes les variables, ou un sous-ensemble de variables, ont un effet monotone connu sur toutes les contraintes et la fonction objectif ou un sous-ensemble de ces fonctions.

1.1 Motivations

Le sujet de cette recherche a été grandement motivé par une problématique rencontrée par les experts en optimisation de la compagnie Rio Tinto lors du déploiement du logiciel d'optimisation NOMAD [40] sur un de leur problème de type boîte noire nommé `Kemano`. NOMAD est une implémentation en C++ de l'algorithme d'optimisation sans dérivée MADS (Mesh Adaptative Direct Search) proposé par Audet et Dennis [8] disponible gratuitement en ligne [1]. La boîte noire `Kemano` est utilisée hebdomadairement pour aider les ingénieurs de Rio Tinto à maximiser la quantité d'énergie produite par leur barrage hydroélectrique tout en gardant le contrôle sur les risques d'inondation des terrains avoisinants. Une description plus approfondie de la boîte noire `Kemano` est présentée au chapitre 4 .

Des expériences numériques, conduites par un ingénieur analyste en recherche opérationnelle chez Rio Tinto [27], suggèrent qu'il est difficile pour NOMAD de trouver une solution réalisable sur le problème `Kemano`. Plus précisément, la problématique soulevée est qu'il est plus efficace de trouver manuellement une solution satisfaisant les contraintes pour ensuite laisser NOMAD raffiner celle-ci. En effet, à partir d'une solution non réalisable quelconque, l'utilisateur est capable de trouver en une dizaine d'évaluations, une solution réalisable contrairement à NOMAD qui peut chercher durant plusieurs heures. Nous avons testé NOMAD, avec ses paramètres par défaut, sur 10 points de départs non réalisables échantillonnés à l'aide d'un hypercube latin [45]. Le tableau 1.1 indique le nombre d'évaluations de la fonction objectif ainsi que le temps de calcul approximatif qu'il a fallu à NOMAD afin de trouver une solution réalisable à partir de ces 10 points de départ. Un «X» dans le tableau indique qu'aucune solution réalisable n'a été trouvée par l'algorithme avant d'atteindre le nombre maximal d'évaluations de la boîte noire fixé à 500.

Tableau 1.1 Performance de NOMAD sur la boîte noire `Kemano` avant de trouver une solution réalisable avec un budget de 500 évaluations

x_0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Nbr. éval.	X	125	170	X	X	X	X	144	271	117
Temps	9h28	2h22	3h13	9h28	9h28	9h28	9h28	2h43	5h08	2h13

Il est vite devenu clair que l'efficacité de la méthode manuelle découle de la connaissance des ingénieurs sur les effets monotones de certaines variables sur `Kemano`. L'hypothèse émise est que, au point de vue du temps de calcul, il serait avantageux de guider les méthodes par recherche directe à l'aide de l'intuition de l'utilisateur. La problématique soulevée a donc permis de tracer la ligne directrice de ce projet : trouver une façon d'inclure des informations

sur la monotonie d'une fonction dans un algorithme d'optimisation par recherche directe tel que MADS. Pour cela, une attention plus particulière sera apportée à l'amélioration des contraintes plutôt que sur l'optimisation de la fonction objectif.

L'intuition de l'utilisateur représente une première façon de récolter des informations sur la monotonie. Dans ce travail, deux autres méthodes seront explorées. Lorsque la forme analytique d'une (ou de plusieurs) fonction est connue, il est plus facile de spéculer sur les effets monotones exacts. Par exemple, $f(x) = \lfloor x \rfloor$ est monotone croissante. L'information, à portée de main, mériterait sûrement d'être exploitée en cours d'optimisation. La dernière méthode explorée est celle de l'échantillonnage. Celle-ci s'avérera plus utile dans un contexte d'optimisation de boîtes noires. Elle consiste à créer un échantillon de plusieurs points pour ensuite bouger une variable à la fois. Bien que plus dispendieuse, en terme de nombre d'évaluations de la boite noire, elle permet de détecter des informations précieuses.

1.2 Plan du mémoire

Ce mémoire est divisé de la façon suivante. Le chapitre 2 se veut une revue de la littérature sur le sujet large de la DFO. Puisque la nature du travail proposé impose une modification ou l'ajout d'une extension à MADS, l'évolution des algorithmes par recherche directe de type recherche par motifs sera présentée. Plus particulièrement, nous devrons passer par des algorithmes tels que la recherche par coordonnée (CS) [33] et la recherche par motifs généralisée (GPS) [56] avant d'en arriver à la description de MADS. Aussi, une discussion approfondie sur les définitions de boîtes grises retrouvées dans la littérature y est incluse. Finalement, comme nous travaillerons dans un contexte sous contraintes, quelques concepts généralement utilisés en optimisation sans contraintes devront être revus, notamment en ce qui a trait la présentation des résultats numériques à l'aide de profils de performance et de données.

Le chapitre 3 entre dans le vif du sujet. Il a pour but de dresser la base théorique sur laquelle reposeront les nouvelles méthodes proposées au chapitre 4 concernant l'exploitation d'une structure monotone par MADS. Plus particulièrement, il sera question de la matrice de tendance T qui permet de transformer les informations sur la monotonie d'un problème en une entité facilement exploitable par un algorithme. Un premier exemple d'utilisation de la matrice T, trop simple et naïve, sert d'introduction au chapitre 3. Il met en lumière les problèmes engendrés par un manque de rigueur théorique envers le concept de la monotonie et dévoile l'importance d'étudier ce dernier. En fait, la définition de la monotonie d'une fonction à plusieurs variables n'est pas unique. Parmi toutes ces définitions non équivalentes retrouvées dans la littérature, nous prendrons le temps de cerner celle qui semble la plus

appropriée pour mener à bien ce projet. Finalement, la présentation d'une nouvelle méthode plus adéquate concernant la construction et l'utilisation de la matrice T conclura ce chapitre.

Contrairement au chapitre 3 plus théorique, le chapitre 4 se penche sur l'aspect pratique du sujet de recherche. Il contient les variantes de MADS qui ont été testées. De plus, trois façons différentes de déterminer le contenu de la matrice de tendance seront explorées : l'analyse directe des équations du problème, l'échantillonnage de points et l'intuition de l'utilisateur de la boîte grise. Finalement, nous testerons ces nouveaux outils sur des problèmes tests académiques et industriels. Une synthèse du projet ainsi qu'une discussion générale sur les résultats obtenus et les travaux futurs se retrouvent au chapitre 5.

1.3 Précision sur le contexte et notation

Avant de passer à la revue de littérature, la présente section permet de préciser la forme théorique générale des problèmes qui seront abordés dans ce travail. La notation introduite servira tout au long de ce document.

Soit le problème de minimisation

$$\begin{aligned} &\min_{x \in X} \ f_0(x) \\ &\quad \text{s.c } f_j(x) \leq 0, \quad j \in \{1, 2, \ldots, m\}, \end{aligned} \tag{1.1}$$

où $X \subseteq \mathbb{R}^n$ et $f_j(x) \leq 0$ avec $j = 1, 2, \ldots, m$ représentent respectivement les contraintes non relaxables et relaxables. Combinées, elles forment l'ensemble des points réalisables $\Omega = \{x \in X : f_j(x) \leq 0 \ \forall j = 1, 2, \ldots, m\}$. Ces deux types de contraintes se distinguent par la permissivité accordée à l'évaluation d'un point violant celles-ci.

Le respect de l'ensemble X est rigide et, dans le cas d'une transgression, pourrait entraîner une erreur fatale dans l'évaluation de la fonction $f(x) : \mathbb{R}^n \mapsto \{\mathbb{R} \cup \{\infty\}\}^{m+1}$ comparable à celle engendrée par le calcul d'une racine carrée dont l'argument est négatif. Selon les règles de classification des contraintes proposées par Le Digabel et Wild [41], X est de type KAUQ. En effet, dans tous les problèmes tests qui seront présentés, X est un hypercube $l_b \leq x \leq u_b$ dont l'existance est connue (Known) et dont la forme algébrique est disponible (Algebraic). Le niveau de satisfaction des contraintes est donc quantifiable (Quantifiable). De plus, comme on ne peut évaluer la boîte noire à l'extérieur de X, l'ensemble est dit non relaxable (Unrelaxable).

Les contraintes relaxables admettent une plus grande flexibilité et peuvent même donner de l'information utile à un logiciel d'optimisation (Relaxable). Par exemple, si un projet

industriel est limité par un budget, les valeurs retournées par $f(x)$ sont fiables. Le niveau de violation et de satisfaction des contraintes est quantifiable (Quantifiable). Selon la nature des problèmes tests qui seront présentés, la forme algébrique des contraintes $f_j(x)$ avec $j = 1, 2, \ldots, m$ pourrait être disponible (Algebraic) ou non (Simulation). Nous supposerons que toutes les contraintes sont connues (Known). C'est-à-dire que, pour toute évaluation de la boîte noire à l'intérieur de X, le programme ne retourne pas une erreur, NaN ou ∞. Toujours selon la classification de Le Digabel et Wild, les contraintes $f_j(x)$ avec $j = 1, 2, \ldots, m$ retrouvées dans les problèmes tests présentés au chapitre 4 seront donc soit de type QRAK ou QRSK.

Les contraintes non relaxables seront traitées avec la méthode de la barrière extrême [8] tandis que la barrière progressive [9] pourra être déployée sur les contraintes relaxables. Une description de ces deux méthodes est donnée dans le chapitre 2. Dans la suite de ce travail, la notation $f(x)$ fera référence aux fonctions définissants le problème (1.1). La première sortie de $f(x)$ est la fonction objectif et les m dernières sont les contraintes. Finalement, aucune hypothèse sur la continuité ou la différentiabilité de $f(x)$ n'est imposée. Il sera donc pris pour acquis que l'évaluation d'un éventuel gradient est inutile ou hors de portée.

CHAPITRE 2 L'OPTIMISATION SANS DÉRIVÉES

Ce chapitre concerne essentiellement les méthodes algorithmiques appliquées à l'optimisation de boîtes noires. Plus particulièrement, nous discuterons de l'évolution des méthodes par recherche directe et présenterons les raisons motivant ce choix de travailler avec les algorithmes de recherche par motifs. De plus, l'absence de consensus sur la définition du terme «boîte grise» dans la littérature sera mise en lumière. Une définition compatible avec le cadre de ce travail sera alors proposée. Puisqu'il sera ultérieurement question d'optimisation sous contraintes à partir d'un point de départ non réalisable, quelques techniques pour gérer les contraintes seront discutées. Finalement, différentes méthodes qui serviront à l'étude comparative des algorithmes sur les problèmes tests ainsi que les modifications qui leur sont imposées par le contexte de l'optimisation sous contraintes seront explorées.

2.1 L'optimisation de boîtes noires

Bien que plusieurs algorithmes pour l'optimisation de boîtes noires aient été proposés depuis 1952, le contenu de cette revue de littérature porte principalement sur les algorithmes d'optimisation locale par recherche directe tels que la recherche par coordonnées CS, la recherche par motifs généralisée GPS et MADS. Un survol plus large des algorithmes d'optimisation sans dérivées a été présenté en 2012 par Rios et Sahinidis [52]. Cet article propose un résumé de méthodes populaires utilisées dans le contexte d'optimisation de boîtes noires sans contraintes. Les auteurs classent les algorithmes sous trois grandes bannières : les méthodes locales/globales, par recherche directe/avec modèles et stochastique/déterministe. Cet article présente aussi une revue historique de l'évolution du domaine de la DFO. Un second survol, publié en 2014 par Audet [6], propose une revue axée sur l'optimisation de boîtes noires ainsi qu'une étude plus approfondie sur la comparaison de différents algorithmes de DFO appliqués à des problèmes industriels. Le sujet large de la DFO est exploré dans deux manuels d'introduction rédigés par Conn, Scheinberg et Vicente [26] ainsi que par Audet et Hare [10].

2.1.1 La recherche par coordonnées (CS)

Hooke et Jeeves [39] sont parmi les premiers à proposer une méthode de type recherche par motifs. Ces derniers décrivent les méthodes par recherche directe comme

> «l'examination sequentielle de candidats par la comparaison avec la meilleure solution obtenue jusqu'à présent ainsi que l'introduction d'une stratégie afin de

déterminer les prochains candidats les plus prometteurs».

L'idée principale de cet algorithme est d'évaluer la fonction objectif autour d'une solution actuelle dans des directions qui forment une base génératrice de l'espace. Ils appellent cette méthode la «recherche par motifs» plus connue sous le nom de «pattern search». Bien que les auteurs laissent la liberté du choix de cette base au lecteur, celle suggérée par Hooke et Jeeves est l'ensemble des vecteurs des axes coordonnées positifs et négatifs $\{\pm e_i \ : \ i = 1, 2, \ldots, n\}$. Cependant, bien que Hooke et Jeeves ne citent pas Fermi et Metropolis [33], ce sont ces derniers qui ont proposé un algorithme très semblable à la recherche par coordonnées CS (Coordinate Search) en 1952 , avant Hooke et Jeeves en 1961. Le pseudo-code de la recherche par coordonnées est donné par l'algorithme 1 présenté ci-dessous.

Algorithme 1 : Recherche par Coordonnées (CS)

Soit une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\delta^0 \in (0, \infty)$ longueur du pas

$\epsilon \in [0, \infty)$ critère d'arrêt

$k \leftarrow 0$ compteur d'itérations

1. Sonde locale

si $f(\bar{x}) < f(x^k)$ *pour un certain* $\bar{x} \in P^k = \{x^k \pm \delta^k e_i \ : \ i \in \{1, 2, \ldots, n\}\}$ **alors**

$x^{k+1} \leftarrow \bar{x}$ **et** $\delta^{k+1} \leftarrow \delta^k$;

sinon

$x^{k+1} \leftarrow x^k$ **et** $\delta^{k+1} \leftarrow \frac{1}{2}\delta^k$;

2. Arrêt

si $\delta^k \geq \epsilon$ **alors**

$k \leftarrow k + 1$ et retourner à **1** ;

sinon

arrêt ;

Seule une justification heuristique de l'optimalité de la solution trouvée est donnée par Hooke et Jeeves. Une des critiques de CS est qu'il est possible que l'algorithme converge vers un point non optimal. En effet, on n'a qu'à choisir la fonction strictement convexe $f_0(x) = \|x\|_\infty$ avec $x^0 = (1, 1)$. Dans ce cas, CS raffinera autour de x^0 jusqu'à ce que le critère d'arrêt soit atteint alors que la solution optimale est $x^* = (0, 0)$. Une seconde critique est que le sujet des problèmes sous contraintes n'a pas été abordé dans l'article et l'algorithme 1 peut difficilement être utilisé sur le problème (1.1).

2.1.2 La recherche par motifs généralisée (GPS)

En 1997, Torczon propose une généralisation des méthodes d'optimisation de type recherche par motifs en les rassemblant sous une structure commune et en donnant une preuve de convergence pour cette famille d'algorithmes [56]. Torczon mentionne notamment la recherche par coordonnées avec des longueurs de pas fixes (Fermi et Metropolis [33]), l'opération évolutionnaire (Box [21]), la recherche par motifs (Hooke et Jeeves [39]) ainsi que l'algorithme de recherche multidirectionnelle (Dennis et Torczon [31]). L'algorithme proposé est la recherche par motifs généralisée que nous abrègerons par GPS (Generalized Pattern Search). Elle montre que la longueur du pas δ_k n'a pas à être monotone décroissante comme dans CS pour satisfaire les conditions d'optimalité et que si la fonction objectif possède des courbes de niveau bornées et est $\mathcal{C}^1$, alors $\liminf_{k\to\infty} \|\nabla f(x^k)\| = 0$. Un résultat plus fort $\lim_{k\to\infty} \|\nabla f(x^k)\| = 0$ peut être obtenu si certaines conditions sont ajoutées sur l'étape de sonde locale et sur la matrice génératrice des directions dans l'étape de la sonde locale. Audet [5] montre que les conditions de convergence de GPS sont serrées : il présente une série d'exemples sur lesquels l'algorithme échoue dès que l'une des conditions est relâchée.

Une nouveauté importante retrouvée dans GPS est qu'il est possible d'ajouter une étape de recherche globale, pouvant user d'une méthode heuristique, avant l'étape de sonde locale. Cet ajout d'une méthode heuristique ne change pas les résultats de convergence et peut servir à s'échapper d'un minimum local. L'algorithme GPS est décrit par le pseudo-code de l'algorithme 2.

GPS est une généralisation de la recherche par motifs proposée par Hookes et Jeeves [39], car les modifications apportées assouplissent l'utilisation de l'algorithme sans changer les résultats de convergence. Tout d'abord, GPS permet au treillis d'élargir/rétrécir d'un facteur $\tau/\bar{\tau}$ dans le cas d'un succès/échec respectivement. Un choix populaire est de poser $\bar{\tau} = \tau^{-1}$. De plus, une recherche globale peut être menée par l'utilisateur, en autant que les points soumis à l'évaluation S^k se retrouvent sur le treillis (en anglais «Mesh»)

$$M^k = \{x^k + \delta^k D y \ : \ y \in \mathbb{N}^p\}.$$

Finalement, l'ensemble des points D^k soumis à l'étape de sonde locale n'a pas besoin d'être statique. C'est-à-dire qu'il peut être changé au cours de l'optimisation, en autant que celui-ci forme un sous-ensemble générateur positif des colonnes de $D = GZ$, où $G \in \mathbb{R}^{n\times n}$ est inversible et $Z \in \mathbb{Z}^{n\times p}$ est une matrice dont les vecteurs colonnes forment un ensemble générateur positif de $\mathbb{R}^n$ avec $p \geq n+1$.

Algorithme 2 : Recherche par motifs généralisée (GPS)

Soit une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\delta^0 \in (0, \infty)$ longueur du pas

$D = GZ$ un ensemble générateur positif

$\tau \in (1, \infty)$ un facteur d'élargissement

$\bar{\tau} \in (0, 1)$ un facteur de rétrécissement

$\epsilon \in [0, \infty)$ critère d'arrêt

$k \leftarrow 0$ compteur

1. Recherche globale

si $f(\bar{x}) < f(x^k)$ *pour un certain* $\bar{x}$ *dans un sous-ensemble fini* S^k *du treillis* M^k **alors**

$x^{k+1} \leftarrow \bar{x}$ **et** $\delta^{k+1} \leftarrow \tau\delta^k$ et aller à **3**;

sinon

aller à **2** ;

2. Sonde locale

Soit un sous-ensemble générateur positif $D^k \subseteq D$

si $f(\bar{x}) < f(x^k)$ *pour un certain* $\bar{x} \in P^k = \{x^k \pm \delta^k d \ : \ d \in D^k\}$ **alors**

$x^{k+1} \leftarrow \bar{x}$ **et** $\delta^{k+1} \leftarrow \tau\delta^k$;

sinon

$x^{k+1} \leftarrow x^k$ **et** $\delta^{k+1} \leftarrow \bar{\tau}\delta^k$;

3. Arrêt

si $\delta^k \geq \epsilon$ **alors**

$k \leftarrow k + 1$ et retourner à **1** ;

sinon

arrêt ;

L'ensemble des colonnes A_j, $j \in \{1, 2, \ldots, p\}$ de la matrice $A \in \mathbb{R}^{n\times p}$ est un ensemble générateur positif si

$$\mathbb{R}^n = \left\{ \sum_{j\in\{1,2,\ldots,p\}} \lambda_j A_j \ : \ \lambda \geq 0 \right\}.$$

De nombreuses définitions et plusieurs résultats théoriques ont été énoncés en 1954 par Davis [28] au sujet des ensembles générateurs positifs. De plus, Regis [51] explore les propriétés de ces ensembles dans le cadre de la DFO. Plus particulièrement, il propose des procédures de construction d'un ensemble générateur positif potentiellement intéressantes aux yeux d'un utilisateur d'algorithmes par recherche directe. Audet et al. [11] se sont aussi penchés sur la

construction d'un ensemble générateur positif minimal afin de diminuer le nombre d'évaluation de la fonction objectif par l'algorithme MADS sans changer les résultats de convergence. Ce dernier est présenté dans la section suivante.

2.1.3 Recherche directe par treillis adaptatif (MADS)

Une des plus grandes limitations de l'algorithme GPS est qu'il n'est pas efficace à l'ajout de contraintes [9], sans oublier que les preuves de convergences fournies ne sont admissibles que pour les problèmes sans contraintes. En 2000, Lewis et Torczon ont prouvé la convergence de GPS sur une fonction objectif différentiable dans le cas de contraintes linéaires connues vers un point stationnaire au sens de Karush-Khun-Tucker [43]. Ils ont adapté l'étape de sonde locale pour que les directions de l'ensemble générateur positif D^k engendrent le cône tangent du domaine réalisable lorsque la solution actuelle se trouve proche de la frontière de Ω. Par contre, la méthode proposée est valide sous l'hypothèse forte que les contraintes ont une forme linéaire connue. Audet et Dennis se sont adressés à ce problème en introduisant la méthode de recherche directe par treillis adaptatif [8] MADS (Mesh Adaptive Direct Search). La différence majeure entre ces deux algorithmes est l'ajout d'un cadre F^k dans lequel l'ensemble des directions candidates à l'étape de sonde locale devient asymptotiquement dense. En fait, cet ensemble D^k_Δ n'a plus besoin de provenir d'une liste fixe de directions et peut être généré de façon très générale en autant qu'elle respecte deux conditions : être une base positive et être sur le treillis M^k qui se raffine plus vite que le cadre F^k. Grâce à cette nouveauté, de meilleurs résultats de convergence reposant sur la dérivée de Clarke [23] et le cône hypertangent [53] ont été obtenus dans le cas d'optimisation sous contraintes. En d'autres termes, si MADS converge vers un minimum local se trouvant sur la frontière de Ω, alors l'algorithme s'adapte à la géométrie locale des contraintes, quelles soient linéaires ou non. Cela assure ainsi la convergence vers le vrai minimum local. La force de ces résultats de convergence est proportionnelle aux conditions imposées sur la fonction objectif et les contraintes (continue, Lipschitz et différentiable). La définition suivante est donnée par Audet et Dennis [8].

Définition 2.1. *(Cadre) Soit $G \in \mathbb{R}^{n\times n}$ une matrice inversible et $Z \in \mathbb{Z}^{n\times p}$ telles que les colonnes de Z forment un ensemble générateur positif de $\mathbb{R}^n$. Soit $D = GZ$ et les paramètres $\delta^k, \Delta^k > 0$ tels que $\delta^k \leq \Delta^k$. Le cadre généré par D centré à la solution actuelle $x^k \in \mathbb{R}^n$ de taille Δ^k est défini par :*

$$F^k := \left\{ x^k + \delta^k d \ : \ d = Dy, \ y \in \mathbb{N}^p, \delta^k \|d\|_\infty \leq \Delta^k \max\{\|d'\| : d' \in D\} \right\}.$$

où δ^k et Δ^k représentent respectivement la taille du treillis et du cadre.

En posant $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$, le treillis se raffine plus rapidement que le cadre, créant ainsi un plus grand choix de points dans F^k.

Comme il est maintenant possible de considérer des contraintes non linéaires, le pseudo-code doit être ajusté à cette nouveauté. L'implémentation de base de MADS donnée par l'algorithme 3 utilise la méthode de la barrière extrême afin de transformer un problème avec contraintes en problème non contraint :

$$f_\Omega(x) := \begin{cases} f(x) & \text{si } x \in \Omega, \\ \infty & \text{si } x \notin \Omega. \end{cases}$$

Algorithme 3 : Recherche directe par treillis adaptatif (MADS)

Soit une fonction $f_\Omega : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ et un point de départ $x^0 \in \Omega$

0. Initialisation

$\Delta^0 \in (0, \infty)$ taille initiale du treillis
$D = GZ$ un ensemble générateur positif
$\tau \in (0, 1)$ un facteur d'élargissement rationnel du treillis
$\epsilon \in [0, \infty)$ critère d'arrêt
$k \leftarrow 0$ compteur

1. Mise à jour des paramètres

Poser la taille du treillis $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$

2. Recherche globale

si $f_\Omega(\bar{x}) < f_\Omega(x^k)$ *pour un certain* $\bar{x}$ *dans un sous-ensemble fini* S^k *du treillis* M^k **alors**
$x^{k+1} \leftarrow \bar{x}$ **et** $\Delta^{k+1} \leftarrow \tau^{-1}\Delta^k$ et aller à **4**;
sinon
aller à **3** ;

3. Sonde locale

Soit un sous-ensemble générateur positif D_Δ^k tel que
$x^k + \delta^k d$ est dans le cadre F^k de précision Δ^k pour tout $d \in D_\Delta^k$
si $f_\Omega(\bar{x}) < f_\Omega(x^k)$ *pour un certain* $\bar{x} \in P^k = \{x^k \pm \delta^k d \ : \ d \in D_\Delta^k\}$ **alors**
$x^{k+1} \leftarrow \bar{x}$ **et** $\Delta^{k+1} \leftarrow \tau^{-1}\Delta^k$;
sinon
$x^{k+1} \leftarrow x^k$ **et** $\Delta^{k+1} \leftarrow \tau\Delta^k$;

4. Arrêt

si $\Delta^k \geq \epsilon$ **alors**
$k \leftarrow k + 1$ et retourner à **1** ;
sinon
arrêt ;

L'algorithme MADS continue d'évoluer. Un ajout récent important est celui des modèles quadratiques dans les étapes de recherche globale (trouver des candidats prometteurs) et de

sonde locale (ordonnancement de P^k) [4, 25]. Audet et Tribes ont aussi obtenu des résultats intéressants à l'ajout de la méthode de Nelder-Mead dans l'étape de la recherche globale [15]. Deux avantages particuliers ont conduit au choix de MADS comme algorithme de base pour ce projet. Tout d'abord, il repose sur une base théorique solide qui lui permet de se débarrasser du statut d'heuristique porté par d'autres méthodes par recherche directe. De plus, son implémentation est flexible, car l'étape de recherche globale peut être librement choisie par l'utilisateur de la boîte noire et l'ordre dans lequel les directions de D_Δ^k sont évaluées pendant la sonde locale ne change pas la théorie de convergence. Un critère déterminé par le concepteur de la boîte noire peut alors indiquer les directions les plus prometteuses à évaluer en premier pour ainsi sauver des évaluations coûteuses de la boîte noire grâce à une stratégie opportuniste.

Le choix de MADS pour ce projet repose sur plusieurs autres motivations extrinsèques. Premièrement, les ingénieurs en optimisation de Rio Tinto Alcan veulent améliorer la performance de l'outil d'optimisation NOMAD [1] qu'ils utilisent sur une de leur boîte noire. Puisque NOMAD est une implémentation de MADS, la nature du problème donné nous dirige donc vers les algorithmes par recherche directe. Aussi, NOMAD est un logiciel libre en C++ distribué en ligne. Il est donc relativement facile d'apporter des modifications et des extensions à celui-ci, d'autant plus que la documentation complète est disponible. Finalement, dans le contexte de la présente recherche, l'intégration des informations fournies par l'utilisateur dans un algorithme tel que MADS demande de passer par les concepts de monotonie d'une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$. Un sujet théorique riche qui, au meilleur de nos connaissances, n'a jamais été exploité dans le cadre de la DFO.

2.2 L'optimisation de boîtes grises

De façon générale, on retrouve les termes «boîtes noires» et «boîtes grises» dans un contexte de modélisation de phénomènes physiques. Selon Whiten [60], une boîte grise «utilise une structure théorique incomplète qu'il faut compléter à l'aide des données expérimentales». L'auteur explique que la structure théorique connue peut grandement varier entre l'estimation de paramètres d'une équation physique connue et la complétion d'une structure faible par des modèles supplémentaires basés sur les données observées. Whiten laisse donc une grande liberté au lecteur en ce qui a trait à la définition de boîte grise. Dans son livre «Practical Grey-box Process Identification», Bohlin [17] explique qu'il y a deux sources d'informations possibles afin d'ajuster un modèle à un phénomène physique : la connaissance du phénomène observé et les données expérimentales recueillies. Lorsque toutes les équations physiques d'une expérience sont connues (premier cas), il ne reste plus qu'à calculer les paramètres de ces

équations. Bohlin y associe le terme de boîte blanche. Lorsque l'utilisateur ne peut rien dire sur le modèle le plus adéquat décrivant le comportement du système (deuxième cas), on parle de boîte noire. Selon Bohlin, une boîte grise est le mélange de ces deux derniers processus afin de déterminer la fonction qui modélise le mieux le système.

Peu d'articles sur l'optimisation par méthodes non-heuristiques de boîtes grises sont présentement disponibles. La plupart des méthodes proposées se situent dans un contexte d'optimisation avec modèles substituts et non dans le cadre de méthodes par recherche directe. En effet, dans leur article de 2016, Boukouvala et Floudas [19] y décrivent un algorithme nommé ARGONAUT (AlgoRithms for Global Optimization of coNstrAined grey-box compUtational problems) qui s'attaque aux problèmes de type boîtes grises. Ils caractérisent ces dernières par «le manque partiel ou total de formule analytique décrivant les contraintes ou la fonction objectif». Dans le cas où l'expression analytique de l'équation d'une contrainte ou de la fonction objectif est inconnue, celle-ci est approximée à l'aide d'un modèle substitut. ARGONAUT se situe dans la classe d'algorithmes de recherche globale à l'aide de modèles [20]. Dans le même ordre d'idée, Bajaj, Iyer et Hassan se sont intéressés à la résolution de boîtes grises à partir d'un point initial non réalisable à l'aide de modèles substituts [16]. Ils définissent une boîte grise comme une boite noire dont la forme analytique d'au moins une contrainte est connue. Eason et Biegler [32] ont aussi proposé une méthode utilisant les régions de confiance. Cette fois, les auteurs utilisent le terme «boîte de verre» pour référer aux problèmes dont les fonctions explicites sont connues et dont les dérivées peuvent être obtenues à l'aide de la différentiation automatique.

Dans le cadre de ce travail, le sens donné au terme boîte grise est bien différent, car nous ne sommes pas concernés par l'estimation de paramètres pour l'ajustement de modèles contrairement aux travaux cités plus haut, ni par la forme analytique des fonctions de la boîte noire. Dans un contexte industriel, il n'est pas rare de rencontrer une boîte noire où il est possible de donner un sens physique aux variables d'entrée et de sorties sans connaître la forme explicite ou générale du système. Le constructeur du programme peut alors prédire l'effet de ces variables sur la croissance ou décroissance de la fonction à optimiser ainsi que les contraintes relaxables. Dans un tel cas, est-il toujours vrai de dire qu'aucune information n'est à portée de main et qu'il s'agit d'une boîte noire ?

Définition 2.2. *Dans ce projet, le terme* ***boîte grise*** *réfère à une boîte noire ayant la forme du problème* (1.1) *dont les informations partielles (ou complètes) sur l'effet de croissance de certaines variables, sur une des contraintes ou la fonction objectif, sont disponibles. En d'autres termes,* $f(x)$ *doit bénéficier d'une structure monotone connue par l'utilisateur.*

En 2012, Liuzzi et Risi [44] se sont aussi penchés sur le sujet. À proprement parler, l'article

ne se situe pas dans un contexte d'optimisation de boîtes noires. Par contre, les auteurs s'attaquent à une question qui s'approche des boîtes grises. Cette fois, les informations connues sont les valeurs exactes de certaines dérivées partielles de la fonction objectif f_0 pour un sous-ensemble fixe de variables. Les auteurs présentent une méthode d'optimisation qu'ils appellent «décomposition par blocs». Ils trient les variables de $x = (y, z)$ en deux catégories : celles dont les dérivées partielles de f_0 peuvent être calculées (z) et celles dont on n'a aucune information (y). Liuzzi et Risi proposent d'utiliser une méthode de recherche par motifs pour optimiser sur les variables y tout en fixant z. Une fois cette étape terminée, ce sont les variables y qui sont fixées afin d'optimiser sur z à l'aide d'une recherche linéaire de type Armijo basée sur l'usage des dérivées. Ils prouvent aussi la convergence de leur méthode vers un point critique du problème. Comme mentionné précédemment, cette méthode ne s'inscrit pas dans le contexte d'optimisation de boîtes grises, car les auteurs émettent l'hypothèse forte que $f_0 \in \mathcal{C}^1$ et que certaines dérivées sont disponibles.

Un article plus pertinent encore pour ce travail a été publié en 2004 par Abramson, Audet et Dennis [2]. Les auteurs utilisent de l'information sur les dérivées afin d'accélérer la vitesse de convergence de GPS lors de la minimisation de $f_0(x)$ où $x \in \Omega = \{x \in \mathbb{R}^n : Ax \leq b\}$ avec la méthode de la barrière extrême. Rappelons que MADS n'a été proposé que deux ans plus tard. Ces auteurs se concentrent seulement sur l'étape de sonde locale de GPS, plutôt que la recherche globale, afin d'exploiter toute l'information donnée par le gradient. Comme D^k de l'algorithme 2 est un ensemble générateur positif de $\mathbb{R}^n$, il existe une direction $d \in D^k$ telle que $v^\top d < 0$ pour tout v non nul donné, en particulier pour $v = \nabla f_0(x)$ ou, par ce qu'ils appellent, une ϵ-approximation des composantes dominantes du gradient. Les auteurs proposent donc d'utiliser cette propriété de D^k en introduisant la notion d'ensemble *élagué* D_k^p où l'exposant p signifie «pruned»

$$D_k^p = \{d \in D^k \ : \ d^\top \nabla f_0(x^k) \leq 0\}$$

dans le cas où le gradient est connu, ou par

$$D_k^p = D^k \backslash \{d \in V^k \ : \ f_0'(x^k; d) > 0\}$$

où V^k est l'ensemble des directions dont la dérivée directionnelle est connue. Grâce à ces nouvelles définitions, les auteurs considèrent l'ensemble élagué des points de la sonde locale

$$P_k^p = \{x^k + \delta^k d \ : \ d \in D_k^p\}.$$

Les ensembles élagués permettent de cibler les directions de succès afin de réduire le plus

possible le nombre d'évaluations grâce à une stratégie opportuniste. Les auteurs montrent qu'il est suffisant d'utiliser une ϵ-approximation des composantes dominantes du gradient afin d'assurer un succès en une seule évaluation de la fonction. En effet, Abramson et al. concluent que si l'approximation du gradient contient des éléments du même signe que le vrai gradient pour les composantes les plus grandes, alors l'approximation élague l'ensemble des directions de recherche à un singleton.

Lorsque seules $\rho \leq n$ dérivées partielles sont disponibles, ils prouvent que, dans le pire des cas, le nombre d'évaluations de la fonction $f_0(x)$ à une itération k passe de $n+1$ à $n+1-\rho$ lorsque D^k est élagué par rapport à l'ensemble V^k.

Bien que cet ouvrage s'approche de la problématique abordée dans ce projet, il reste que les preuves fournies dans l'article sont données sous la condition que la fonction est différentiable au point d'intérêt. Comme nous travaillerons avec des conditions plus faibles sur la fonction objectif, il ne sera pas possible de prouver que l'on peut élaguer D^k à un singleton contrairement à ce qui a été présenté plus haut. Cependant, la construction du vecteur de descente présentée dans le prochain chapitre a l'avantage d'être indépendante de la forme du gradient. C'est-à-dire qu'il ne sera pas nécessaire de déterminer quelles sont les composantes dominantes du gradient s'il existe.

2.3 Optimisation sous contraintes

Le contexte d'optimisation de boîtes noires implique qu'aucune condition n'est posée sur la fonction objectif et les contraintes, excepté qu'elles retournent des informations utiles pour un point non réalisable $x_{\text{inf}} \notin \Omega$ et non des sorties telles que NaN ou ∞. Cela nous assure que la violation des contraintes est quantifiable par une fonction $h(x)$ à minimiser. Cette dernière sera définie plus loin.

Il existe plusieurs façons de traiter la violation des contraintes. Une première est d'utiliser la méthode de la barrière extrême. Celle-ci a déjà été présentée dans la section précédente. Cette méthode est utile lorsqu'un point de départ réalisable est connu. Plusieurs efforts ont été déployés afin d'adapter les méthodes par recherche directe aux problèmes sous contraintes traitées par la barrière extrême.

Une seconde méthode est celle de la barrière progressive. Cette dernière, introduite et perfectionnée par Audet et Dennis dans [7] et [9], est une modification de la notion de filtre originalement proposée par Fletcher et Leyffer [34]. Elle est implémentée dans NOMAD et servira donc lors des tests numériques présentés au chapitre 4. La caractéristique principale de la barrière progressive est qu'elle permet de rechercher un point réalisable tout en portant

attention à la valeur de la fonction objectif à l'extérieur de Ω. Il est ainsi possible de démarrer l'algorithme MADS sur un point de départ non réalisable.

Peu importe la méthode choisie, lorsque seule une solution non réalisable est disponible, un effort considérable doit être déployé au début de l'optimisation afin de trouver une solution réalisable. Introduisons la fonction $h(x)$ qui mesure la violation des contraintes du problème (1.1) :

$$h(x) = \sum_{j=1}^{m} (\max\{f_j(x), 0\})^2 \tag{2.1}$$

Si aucune solution réalisable n'est connue, alors une première méthode grossière consiste à choisir un point quelconque dans l'ensemble X et transformer le problème (1.1) en une minimisation de $h(x)$ sur $x \in X$ sans se soucier de la fonction objectif $f_0(x)$. La fonction $h(x)$ est intéressante dans le cadre de méthodes par recherche directe, car elle a deux propriétés importantes. Tout d'abord, elle est différentiable lorsque $f_j \in \mathcal{C}^1$ pour tout $j = 1, 2, \ldots, m$. Cela permet d'obtenir des résultats de convergence plus forts sur la solution optimale trouvée par MADS. De plus, $h(x) = 0$ si et seulement si $x \in \Omega$. Il en découle ainsi un critère simple indiquant si x est une solution réalisable ou non au problème (1.1).

Une seconde approche plus sophistiquée est celle de la barrière progressive mentionnée plus tôt. Elle consiste à introduire un seuil décroissant h^k_{max} mis à jour à chacune des itérations. Lorsque la violation des contraintes associées à un point dépasse ce seuil, celui-ci est automatiquement rejeté. L'intérêt d'un seuil décroissant h^k_{max} est qu'il permet de tenir compte de la fonction objectif en autant qu'une condition sur h, de plus en plus sévère, soit respectée. Une description plus détaillée de la barrière progressive, ainsi que le pseudo-code, est proposée par Audet et Hare [10]. Soit x_{inf} l'itéré courant non réalisable. Il est le point qui a obtenu la meilleure valeur de la fonction objectif tout en satisfaisant $h(x_{inf}) \leq h^k_{max}$ depuis le début de l'algorithme d'optimisation. Soit $t \in P^k$ un candidat à l'étape de sonde locale. Il y a trois situations possibles :

- Si t domine x_{inf}, c'est-à-dire que $h(t) \leq h(x_{inf})$ et $f(t) \leq f(x_{inf})$ avec au moins une inégalité stricte, alors x_{inf} est remplacé par t et la barrière h^{k+1}_{max} est diminuée à $h(x_{inf})$;
- Si la solution t améliore seulement la violation des contraintes $h(t) < h(x_{inf})$, alors seule la barrière h^{k+1}_{max} est mise à jour par $h^{k+1}_{max} = \max\{h(v) : h(v) < h(x_{inf}), v \in V\}$, où V est l'ensemble des points évalués jusqu'à présent ;
- S'il n'y a aucune amélioration, alors $h^{k+1}_{max} = h(x_{inf})$.

Lorsqu'une solution réalisable est trouvée, NOMAD continue l'optimisation, avec l'algorithme MADS, de la fonction objectif autour de deux itérés courants (primaire et secondaire) où l'un

est réalisable et l'autre est le point non réalisable avec la meilleure valeur de la fonction objectif trouvée. Il y a un plus grand effort déployé à l'étape de sonde locale autour du point primaire contrairement au point secondaire où l'ensemble des directions de recherche est de plus petite cardinalité et ne forme pas toujours une base positive.

La différence fondamentale entre la barrière extrême et la barrière progressive, présentées ci-haut, est que la première méthode cherche à trouver une solution réalisable le plus rapidement possible, peu importe la détérioration induite sur $f_0(x)$. Quant à la deuxième méthode, elle tient compte de la fonction objectif $f_0(x)$ lors de la minimisation de h.

Dans le cadre de ce travail, un problème concernant certaines définitions couramment utilisées en optimisation sans contraintes a été rencontré. Dans le premier chapitre du manuel «Introduction to Derivative-Free Optimization» [26], Conn, Scheinberg et Vicente définissent d comme une direction de descente de g en x s'il existe $\bar{\alpha} > 0$ tel que $g(x+\alpha d) < g(x),\ \forall \alpha \in]0, \bar{\alpha}]$. Cependant, comme il est expliqué dans le treizième chapitre du même livre, dans le cas de l'optimisation sous contraintes, il est possible qu'une direction de descente soit non réalisable. Cela se produit lorsque l'itéré courant x se trouve sur $\partial\Omega$, la frontière du domaine réalisable. Il est donc nécessaire de définir le concept de direction de descente dans un contexte sous contraintes. Nous en profiterons aussi pour adoucir la condition d'inégalité stricte afin d'accepter un comportement de type escalier.

Définition 2.3. *Soit $x \in X \subseteq \mathbb{R}^n$. Une **direction** $d \in \mathbb{R}^n$ est dite **réalisable** s'il existe $\bar{\alpha} > 0$ tel que $x + \alpha d \in X$ pour tout $\alpha \in]0, \bar{\alpha}]$. L'ensemble des directions réalisables est noté $R_X(x)$.*

Définition 2.4. *Soit $g : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$. Alors $d \in R_X(x)$ est une **direction de descente** pour g en x s'il existe $\alpha' > 0$ tel que $g(x + \alpha d) \leq g(x)$ lorsque $\alpha \in]0, \alpha']$.*

*De la même façon, on définie une **direction de montée** en inversant le signe de l'inégalité $g(x + \alpha d) \geq g(x)$.*

2.4 Méthodes servant d'étude comparative

Dans le chapitre 4, il sera question de la performance de différents algorithmes sur des problèmes tests choisis. Une des difficultés rencontrées est la représentation des résultats servant à la comparaison des performances. Plusieurs outils peuvent être utilisés, tels que : des tableaux, des comparaisons de quartiles ou de moyennes, ordonnancement avec des rangs, etc. Cependant, dans leur article [30], Dolan et Moré déplorent l'efficacité de ces méthodes dans le contexte où l'ensemble de problèmes tests est large et disparate (problèmes faciles ou très difficiles). Parmi les nombreuses lacunes soulevées, on y retrouve : la perte d'informations

quantitatives telles que la taille de l'amélioration sur la fonction objectif, la difficulté de comparer plus de deux algorithmes simultanément, des résultats grandement influencés par un petit ensemble de problèmes tests difficiles, etc. Dans ce même article, Dolan et Moré introduisent une méthode de représentation des résultats nommée profil de performance. Moré et Wild [47] considèrent cette méthode comme une bonne façon de comparer le comportement à long terme des algorithmes étudiés, ce qui n'est pas toujours souhaitable dans un contexte où les boîtes noires sont lourdes à évaluer en un point. Ils proposent alors les profils de données afin d'étudier le comportement à court terme de ces algorithmes. Puisque ces deux méthodes seront utilisées dans ce projet, celles-ci seront explorées dans cette section. De plus, comme les simulations débuteront sur des points de départ non réalisables, quelques modifications devront être apportées afin de complémenter les techniques originales.

2.4.1 Les profils de performance

Moré et Wild [47] s'intéressent aux problèmes d'optimisation sans contraintes $\min\{g(x) : x \in \mathbb{R}^n\}$. Afin de déterminer si un problème connait un succès ou un échec, les auteurs proposent le test de convergence suivant :

$$g(x^0) - g(x) \geq (1 - \tau)(g(x^0) - g^L) \tag{2.2}$$

où x^0 est le point de départ de tous les algorithmes, g_L est la meilleure solution connue du problème et $\tau > 0$ est un paramètre de précision. Le choix d'un plus grand τ indique que l'on accepte une résolution plus grossière du problème, contrairement à un petit τ qui demande à l'algorithme une meilleure approximation de la solution optimale. Bien qu'une solution exacte $g(x^*)$ peut être disponible, Moré et Wild mentionnent qu'il est préférable de choisir g^L comme étant le minimum atteint par l'ensemble des algorithmes testés à partir de x^0. En effet, dans le cas ou le nombre d'évaluations de la fonction est limité, il est possible qu'aucun algorithme ne s'approche suffisamment de $g(x^*)$ pour un τ donné. Dans le cas d'optimisation sous contraintes, le choix de $g(x_0)$ sera discuté plus loin.

Soit $\mathcal{A}$ l'ensemble des algorithmes étudiés et $\mathcal{P}$ l'ensemble des problèmes tests. L'idée principale des profils de performance est de comparer la vitesse relative $r_{a,p}$ qu'il faut à un algorithme $a \in \mathcal{A}$ pour atteindre un succès sur un problème $p \in \mathcal{P}$ par rapport au plus rapide de l'ensemble $\mathcal{A}$. Bien que la vitesse relative peut être calculée en temps d'exécution du programme, il est souhaitable de comparer le nombre d'évaluations de la fonction lorsque celle-ci est longue à évaluer. Notons $N_{a,p}$ le nombre d'évaluations minimal qu'il a fallu à

l'algorithme a avant de satisfaire le critère de convergence (2.2) sur un problème p :

$$r_{a,p} = \frac{N_{a,p}}{\min\{N_{a,p} : a \in \mathcal{A}\}}.$$

Dans le cas où un algorithme n'a pas réussi à atteindre un succès selon le test de convergence (2.2), alors la convention $r_{a,p} = \infty$ est utilisée. Seul l'algorithme le plus rapide pour atteindre un succès sur un problème p obtient une vitesse relative $r_{a,p} = 1$.

Le profil de performance d'un algorithme $a \in \mathcal{A}$ sur un ensemble de problèmes $\mathcal{P}$ en fonction de $\alpha \geq 1$ est donné par l'équation suivante :

$$\rho_a(\alpha) = \frac{1}{|\mathcal{P}|} \times |\{p \in \mathcal{P} : r_{a,p} \leq \alpha\}|. \tag{2.3}$$

Le profil de performance $\rho_a(\alpha)$ a la signification suivante : si on donne à l'algorithme a un budget d'évaluations de la fonction objectif α fois plus élevé qu'à l'algorithme le plus rapide, quel pourcentage de problèmes a est-il capable de résoudre ? Donc $\rho_a(\alpha)$ est borné entre 0 et 1. De plus, comme $\rho_a(1)$ indique la proportion de problèmes sur laquelle l'algorithme a a été le plus rapide avant de satisfaire le test de convergence (2.2), alors $\sum_{a \in \mathcal{A}} \rho_a(1) = 1$.

2.4.2 Les profils de performance sous contraintes

Puisque Moré et Wild ont originairement proposé les profils de performance pour des classes de problème $\mathcal{P}$ sans contraintes [47], il faut demeurer prudent lorsque cette méthode est appliquée sur des algorithmes tel que MADS qui peuvent traiter les problèmes sous contraintes tels que (1.1). En effet, un choix subjectif de $g(x_0)$ de l'équation (2.2) entraînera une signification différente de $\rho_a(1)$. Nous nous adresserons à ces difficultés dans cette section. Posons $h(x)$, la fonction quantifiant la violation des contraintes définie dans la section précédente par l'équation (2.1). Puisque, dans cette section, nous référerons au problème (1.1), la notation $g(x)$ sera remplacée par $f_0(x)$ afin de représenter la fonction objectif.

Dans le cas sous contraintes, une condition supplémentaire à la réalisation d'un succès par un algorithme a est d'arriver à une solution réalisable où $h(x) = 0$. Il est donc possible que, pour un point de départ x^0, la valeur de $f_0(x)$ se détériore au profit de $h(x)$, ce qui entraine une suite de points $\{x^k\}_k$ où $\{f_0(x^k)\}_k$ n'est pas monotone décroissant. Il est donc insensé d'utiliser le test de convergence (2.2) sur $f_0(x)$ tant que $h(x) \neq 0$. De plus, supposons que deux algorithmes $a, b \in \mathcal{A}$ génèrent des points dans le domaine réalisable $h(x) = 0$, un nouveau problème survient lors de l'étude de l'amélioration de $f_0(x)$. En effet, il est grandement probable que les deux algorithmes accèdent au domaine réalisable à partir de deux points

différents. Les questions qui s'imposent sont les suivantes. Comment tracer des profils de performance si les premiers points réalisables diffèrent ? Quelles sont les conséquences sur la signification des profils de performance produits ?

Une façon de traiter le cas sous contraintes est de considérer l'amélioration de f_0 seulement lorsque $h(x) = 0$ et de modifier le critère de convergence (2.2) afin que $f_0(x^0)$ soit remplacé par la plus grande valeur de $f_0(x)$ à l'entrée des algorithmes dans le domaine réalisable, noté $f_0^*(x^r)$. Soit x_a^r le premier point réalisable généré par l'algorithme $a \in \mathcal{A}$. Dans le cas où un algorithme $a \in \mathcal{A}$ ne trouve jamais de solution réalisable, la convention choisie est $f_0(x_a^r) = \infty$.

$$f_0^*(x^r) = \max_{a \in \mathcal{A}} \{f_0(x_a^r) : f_0(x_a^r) \neq \infty\}.$$

Le critère de convergence (2.2) devient

$$f_0^*(x^r) - f_0(x) \geq (1 - \tau)(f_0^*(x^r) - f_0^L). \tag{2.4}$$

La situation est représentée à la figure 2.1.

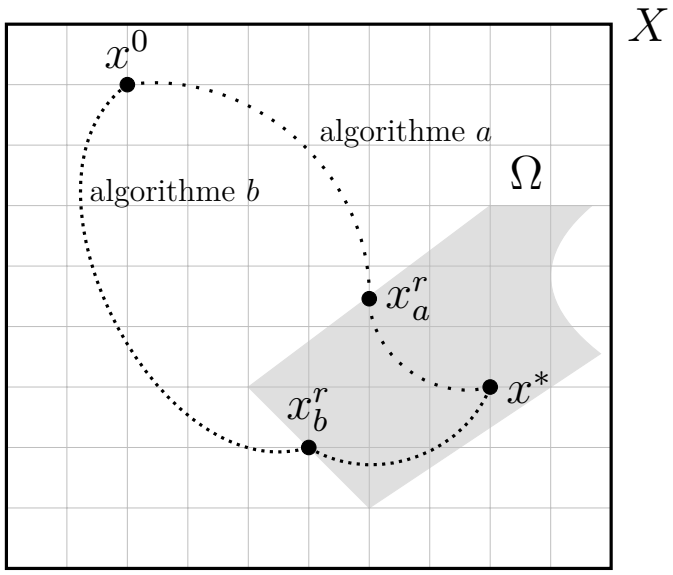


Figure 2.1 Suites de points générés par différents algorithmes à partir d'un point de départ x^0 non réalisable

De ce choix de $f_0^*(x^r)$, il découle qu'un algorithme ayant un meilleur point de départ réalisable est avantagé. Ce choix a été fait pour récompenser l'effort déployé par certains algorithmes qui tiennent compte des valeurs de la fonction objectif pendant l'amélioration de h. Par exemple,

si on considère utiliser une méthode de barrière progressive, il est normal de mettre en valeur l'algorithme qui a investi un effort supplémentaire pour ne pas trop empirer $f_0(x)$ lors de l'amélioration de $h(x)$. Il est possible de poser $f_0^*(x^r)$ autrement, notamment en prenant la moyenne des $f_0(x_a^r)$ sur $a \in \mathcal{A}$ plutôt que le maximum. C'est ce choix qui a été fait par Audet, Le Digabel et Tribes [14] afin d'étudier la performance de l'algorithme MADS à l'ajout d'un échelonnement dynamique sur le treillis. Comme cette décision n'est pas justifiée dans l'article, nous choisirons le maximum plutôt que la moyenne pour la suite de ce projet afin que le côté gauche de l'inégalité (2.4), représentant l'amélioration de la fonction, reste positif.

Puisqu'il est possible que les algorithmes n'arrivent pas tous au domaine réalisable, il résulte que $\sum_{a\in\mathcal{A}} \rho_a(1) \leq 1$. L'égalité retrouvée dans la méthode originale est donc remplacée par une inégalité. En effet, $\rho_a(1)$ nous indique la proportion de problèmes sur lequel l'algorithme a est le plus rapide à résoudre, dans le cas où il trouve une solution réalisable. Cependant, le profil de performance sur $f_0(x)$ n'indique pas la proportion exacte de problèmes où un algorithme a obtenu $h(x) = 0$. Il faut donc compléter l'information obtenue à l'aide d'un deuxième profil de performance sur $h(x)$.

Comme tous les algorithmes démarrent du même point de départ lors de l'amélioration de $h(x)$, le critère de convergence (2.2) peut être utilisé. Cependant, un choix de $h^L = 0$ est plus approprié. La facteur de précision $\tau \geq 0$ peut maintenant prendre la valeur nulle afin d'indiquer le nombre exact de problèmes sur lesquels un algorithme a trouvé une solution réalisable.

Cette version des profils de performance sera utile dans la présentation des résultats au chapitre 4. Néanmoins, il faut demeurer prudent lors de l'analyse de ces graphiques. En effet, comme le soulignent Gould et Scott [35], les profils de performance indiquent quel est l'algorithme qui a le plus de chance de réussir sur l'ensemble de problèmes $\mathcal{P}$, mais ne permet pas de conclure sur la performance relative entre deux algorithmes qui ne sont pas les meilleurs. Les auteurs montrent, qu'au retrait de l'algorithme dominant dans l'étude de performance, les autres courbes peuvent changer d'ordre sur le graphique. Cela découle du changement dans le calcul de $r_{a,p}$. Il sera possible de pallier à ce défaut des profils de performance grâce aux profils de données.

2.4.3 Les profils de données

Moré et Wild [47] définissent les profils de données par la fonction suivante :

$$d_a(\alpha) = \frac{1}{|\mathcal{P}|} \times \left| \left\{ p \in \mathcal{P} : \frac{N_{a,p}}{n_p + 1} \leq \alpha \right\} \right|. \tag{2.5}$$

où $a \in \mathcal{A}$ et n_p est la dimension du problème $p \in \mathcal{P}$.

Les profils de données diffèrent des profils de performance sur deux aspects. Tout d'abord, l'équation (2.5) présente les données brutes et non les performances relatives. De plus, la dimension d'un problème est tenue en compte par le dénominateur $n_p + 1$. Il est normal de s'attendre à ce qu'un problème de dimension plus grande soit plus «long» à optimiser. Diviser $N_{a,p}$ par la taille du simplexe $n_{a,p} + 1$ permet de comparer les problèmes de façon plus égalitaire.

CHAPITRE 3 EXPLOITATION D'UNE STRUCTURE MONOTONE

L'objectif de ce chapitre est de rigoureusement introduire la matrice de tendance T qui contient les informations de monotonie fournies par l'utilisateur et ainsi que d'énoncer les résultats théoriques concernant l'utilisation de cette dernière. Avant toute chose, la section 3.1 présente une méthode simple et naïve d'aborder le problème. Les conséquences du manque de formalisme sur la construction de cette première matrice de tendance T seront mis en lumière afin de justifier le contenu de la section 3.2. Cette dernière est dédiée à l'introduction de quelques définitions de la monotonie d'une fonction $g : \mathbb{R}^n \mapsto \mathbb{R}$ retrouvées dans la littérature. Par la suite, la définition de la matrice T sera révisée et corrigée et ses conséquences théoriques plus fortes seront discutées. Dans la section 3.3, il est montré comment cette nouvelle construction de T vient pallier aux problèmes rencontrés lors de la première approche décrite à la section 3.1.

3.1 Première approche

Pour le moment, aucune hypothèse sur la différentiabilité ou la continuité n'est imposée sur la fonction objectif et les contraintes du problème (1.1). Par contre, seules les situations où il est possible de prédire l'impact d'au moins une variable sur le sens de la croissance de l'une ou plusieurs des fonctions f_j, avec $j \in J = \{0, 1, \ldots, m\}$ seront étudiées. La représentation la plus simple et intuitive de ces informations se fait de la façon suivante. Considérons le cas où l'utilisateur fournit une matrice de tendance T de dimension $n \times (m+1)$ où la première colonne est associée à la fonction objectif et les autres aux m contraintes. Les lignes de T sont associées aux n variables du problème :

$$T_{i,j} = \begin{cases} 1 & \text{si } f_j \text{ augmente lorsque } x_i \text{ augmente} \\ -1 & \text{si } f_j \text{ diminue lorsque } x_i \text{ augmente} \\ 0 & \text{si absence de monotonie ou information inconnue} \end{cases} \tag{3.1}$$
$$\forall j \in J = \{0, 1, \ldots, m\}, i \in \{1, \ldots, n\}.$$

Notons T_j le$j^{\text{ième}}$ vecteur colonne de T.

Supposons qu'un algorithme par recherche directe tel que MADS est utilisé afin de trouver une solution optimale au problème (1.1). Nous voulons utiliser les informations contenues dans la matrice T afin d'accélérer l'étape de sonde locale en utilisant la stratégie opportuniste dans

le cas où le point de départ $x^0 \in X$ est non réalisable (i.e $x^0 \notin \Omega$). Une méthode simple pour s'approcher du domaine réalisable serait d'additionner les colonnes de T associées aux contraintes violées afin d'obtenir une direction $d_T \in \mathbb{R}^n$. Ce vecteur peut être vu comme une direction grossière sur lesquelles les contraintes violées augmentent. Par la suite, on ordonne les directions proposées par MADS à l'étape de sonde locale par rapport à l'angle qu'ils font avec $-d_T$. Plus l'angle est petit, plus il y a de chance que le vecteur soit une direction de descente pour h.

L'exemple 3.1 permet de visualiser cette méthode sur un problème d'optimisation dans $\mathbb{R}^2$. Il sera utilisé à plusieurs reprises dans la suite de ce travail afin de souligner les différences entre les méthodes explorées.

Exemple 3.1. *Considérons le problème de minimisation suivant :*

$$\min_{0 \leq x} \ f_0(x) = x_1^2$$
$$s.c. \ x \in \Omega$$

où le domaine réalisable Ω est défini par les cinq contraintes

$$\begin{aligned}
f_1(x) &= x_2 - 1 - \frac{3}{4}x_1 \leq 0 \\
f_2(x) &= \frac{x_2}{2} - \frac{5}{4} \leq 0 \\
f_3(x) &= -x_2 + 1 - x_1 \leq 0 \\
f_4(x) &= -x_2 - \frac{2}{3} + \frac{2}{3}x_1 \leq 0 \\
f_5(x) &= -\left(x_2 - \frac{23}{12}\right)^2 - \frac{5}{2} + x_1 \leq 0.
\end{aligned}$$

Le domaine Ω est représenté par la zone ombragée dans la figure 3.1.

Comme f_1, f_2, f_3 et f_4 sont des contraintes linéaires, il est facile de vérifier la relation entre chaque variable et chacune de ces fonctions. Par exemple, dans le cas où x_2 est fixé, une augmentation de x_1 entraine automatiquement une diminution de f_1. On pose alors

$$T_{1,1} = -1.$$

Il faut porter une attention particulière à la fonction quadratique f_5 dont la variation reste de même signe par rapport à x_1, mais pas en x_2. En effet, à l'augmentation de x_2, la fonction

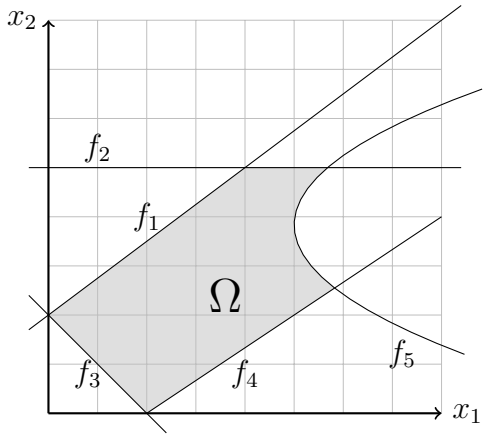


Figure 3.1 Représentation des contraintes de l'exemple 3.1

f_5 croît lorsque $x_2 \leq \frac{23}{12}$ et décroît si $x_2 > \frac{23}{12}$. Dans un tel cas, on posera

$$T_{2,5} = 0.$$

Comme la fonction $f_0(x)$ est elle aussi quadratique, une situation similaire pourrait survenir. Par contre, le domaine Ω se trouve sur le quadrant positif et assure qu'une augmentation de $x_1 \in [0, \infty[$ entraine nécessairement une augmentation de la fonction objectif. Il nous est donc possible de poser $T_{1,0} = 1$.

La matrice de tendance T complète est donnée par :

$$T = \begin{pmatrix} 1 & -1 & 0 & -1 & 1 & 1 \\ 0 & 1 & 1 & -1 & -1 & 0 \end{pmatrix}.$$

Considérons un point de départ x^0non réalisable violant les contraintes f_2 et f_5. L'étape de sonde locale de MADS génèrera un ensemble de directions formant une base positive $D^0 = \{d_1, d_2, d_3, d_4\}$ dans lesquelles la fonction objectif et les contraintes seront évaluées autour de x^0. La situation est présentée à la figure 3.2. Sachant que les contraintes f_2 et f_5 sont violées par x^0, l'idée est d'ordonner judicieusement l'évaluation des directions afin de tirer profit le plus rapidement possible d'une stratégie opportuniste. La méthode proposée dans cette section est d'additionner les colonnes de la matrice T associées aux contraintes violées :

$$d_T = T_2 + T_5 = (0, 1)^\top + (1, 0)^\top = (1, 1)^\top.$$

Par la suite, la liste de directions candidates est ordonnée par rapport à $-d_T$ afin que d_1 et

d_2 soient évaluées avant d_3 et d_4, s'approchant ainsi du domaine réalisable par l'amélioration d'au moins une des deux contraintes.

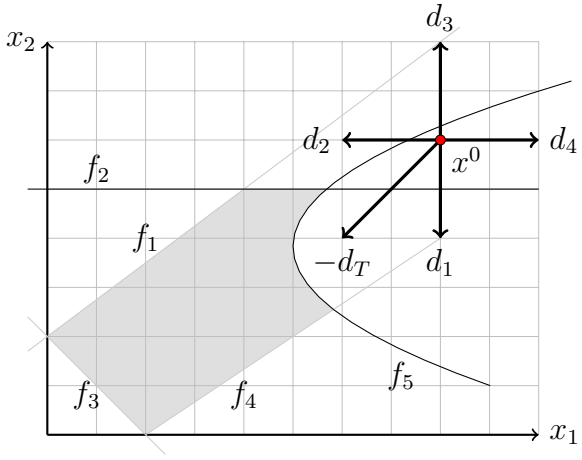


Figure 3.2 Étape de sonde locale autour de $x^0 = (4, 3)^\top$ pour l'exemple 3.1

Bien que visuellement la méthode semble acceptable, il reste que les résultats théoriques qui en découlent sont très faibles, voire inexistants. Tout d'abord, une première partie du problème provient du fait que le calcul de d_T dépend de l'échelle de chacun des vecteurs lignes de T. De plus, même dans le cas où toutes les fonctions $f_j,\ j \in \{0, 1, \ldots, m\}$ sont différentiables, il n'y a aucune garantie que $-d_T$ soit une direction de descente sur $h(x)$, la fonction qui quantifie la violation des contraintes. En effet, supposons que $f_j(x) \in \mathcal{C}^1$, pour tout $j \in \{0, 1, \ldots, m\}$. Par construction de T dans (3.1), on a que $\nabla f_j(x)^\top(-T_j) \leq 0$. Comme cette section se veut une introduction pour la suite du chapitre, nous resterons informels pour l'instant sur cette dernière inéquation et le lien entre T_j et le gradient de f_j sera précisé dans la prochaine section.

Soit $J(x) = \{j \in \{1, 2, \ldots, m\} \ : \ f_j(x) \geq 0\}$ l'ensemble des indices des contraintes actives en x et $\bar{x}$ une solution non réalisable, alors

$$\begin{aligned} h(\bar{x}) &= \sum_{j=1}^{m} \max\{f_j(\bar{x}), 0\}^2 \\ &= \sum_{j \in J(\bar{x})} f_j(\bar{x})^2 \\ \Rightarrow \nabla h(\bar{x}) &= 2 \sum_{j \in J(\bar{x})} f_j(\bar{x}) \nabla f_j(\bar{x}). \end{aligned}$$

On peut vérifier que $-d_T(\bar{x}) = -\sum_{j \in J(\bar{x})} T_j$ est une direction de descente pour $h(\bar{x})$ à l'aide

du produit scalaire suivant :

$$\nabla h^\top(\bar{x})(-d_T(\bar{x})) = \left(2\sum_{j\in J(\bar{x})} f_j(\bar{x})\nabla f_j^\top(\bar{x})\right)\left(\sum_{j\in J(\bar{x})} -T_j\right).$$

En raison de la distribution induite par le produit scalaire, on ne peut pas garantir que la valeur de ce dernier est inférieure ou égale à 0. Même si $\nabla f_j^\top(\bar{x})(-T_j) \leq 0$, il n'est pas assuré que $\nabla f_i^\top(\bar{x})(-T_j) \leq 0$ lorsque $j \neq i \in J(\bar{x})$. L'exemple précédent illustre bien l'idée. Même si la direction $-d_T = -T_2 - T_5$ semble nous approcher du domaine réalisable sur la figure 3.2, il est possible de montrer qu'elle n'est pas une direction de descente pour $h(x^0)$.

Pour x dans un voisinage de $x^0 = (4,3)^\top$, on a $h(x) = f_2(x)^2 + f_5(x)^2$ et

$$\begin{aligned}\nabla h^\top(x)(-d_T) &= (2f_2\nabla f_2^\top + 2f_5\nabla f_5^\top)(-T_2 - T_5)\\ &= -2f_2\nabla f_2^\top T_2 - 2f_5\nabla f_5^\top T_2 - 2f_2\nabla f_2^\top T_5 - 2f_5\nabla f_5^\top T_5.\end{aligned}$$

En évaluant la dérivée directionnelle au point $x^0 = (4,3)^\top$

$$h'(x^0, -d_T) = \nabla h^\top(4,3)(-d_T) = -\frac{1}{4} + \frac{611}{432} - 0 - \frac{47}{72} = \frac{221}{432} > 0.$$

En conclusion, $-d_T$ n'est pas une direction de descente en x^0, car même si les premier et dernier termes sont négatifs comme prévu, il reste que le deuxième ne l'est pas. Le problème est que la construction de d_T ne permet pas d'assurer des termes négatifs lors de la distribution du produit scalaire.

Une façon de pallier à ce problème serait de pondérer la matrice T selon la force d'influence de chaque $f_j(x), j \in J$ sur la fonction $h(x)$. En d'autres termes, si une fonction f_j varie beaucoup, on peut croitre la taille des éléments sur la colonne T_j. On augmenterait alors la probabilité que les termes négatifs du produit scalaire dominent lors de l'addition des termes. Dans l'exemple précédent, poser $T_5 = (2,0)^\top$ plutôt que $(1,0)^\top$ aurait suffit pour rendre le produit scalaire négatif. Il faudrait en faire de même avec la sensibilité des fonctions par rapport à leurs n variables. Comme nous travaillons dans un contexte sans dérivées, l'estimation des poids des $n \times (m+1)$ éléments de T résulterait en une importante étude statistique nécessitant l'échantillonnage d'un grand nombre de points, contrevenant ainsi à la nature même de l'optimisation de boîtes noires. Cependant, Adjengue, Audet et Ben Yahia [3] ont montré que l'utilisation d'une méthode statistique pour identifier les variables importantes peut être avantageuse lorsque MADS est lancé sur des problèmes de grandes dimensions (250 variables ou plus), ce qui n'est pas le cas dans le cadre de ce travail. De plus, même si un tel

effort était déployé afin de calculer la sensibilité des variables et des contraintes, il y aurait encore absence de preuve analytique sur l'amélioration de la fonction $h(x)$.

Finalement, même dans le cas où la matrice T est pondérée adéquatement, un autre problème survient lorsque l'algorithme par recherche directe raffine autour d'une solution optimale x^* qui se trouve sur la frontière des contraintes. En effet, il n'est pas évident de déterminer s'il est plus avantageux de mettre de l'énergie sur le respect des contraintes ou sur l'amélioration de la fonction objectif f_0, en particulier si T_0 est de signe contraire aux autres colonnes de T associées aux contraintes actives. Dans un tel cas, un poids $\xi > 0$ devrait être posé, donnant ainsi une importance plus grande (ou plus petite) à l'amélioration de f_0 par rapport au respect des contraintes pour le calcul de d_T. Il en résulterait ainsi une méthode paramétrée par un choix subjectif de ξ :

$$d_T(x) = \xi T_0 + \sum_{j \in J(x)} T_j.$$

En fait, la construction de T dans (3.1) et le calcul de d_T échouent dans l'exploitation de la structure monotone d'une fonction qui est pourtant extrêmement riche en information. Il existe une meilleure méthode pour approcher ce problème afin d'outrepasser le statut d'heuristique. Avant d'introduire un ajustement pour les constructions de T et de d_T dans la section 3.3, il faut passer par une étude approfondie de la monotonie d'une fonction $g : \mathbb{R}^n \mapsto \mathbb{R}$ afin d'éviter toutes les embûches rencontrées dans la présente section. À l'ajout d'une condition de semi-continuité supérieure qui sera définie plus loin, il sera possible de montrer que $-d_T(x)$ est une direction de descente sur $h(x)$ au sens de la définition 2.4.

3.2 Monotonie dans $\mathbb{R}^n$

Le concept de monotonie pour une fonction $g : \mathbb{R} \mapsto \mathbb{R}$ est bien connu et défini. Par contre, comme le soulignent Van Dyke, Vixie et Asaki [58], lorsque $g : \mathbb{R}^n \mapsto \mathbb{R}$, il n'existe pas une unique définition au sein de la communauté mathématique. En effet, ils énoncent plusieurs définitions retrouvées dans la littérature et montrent que celles-ci ne sont pas toutes équivalentes. Par exemple, on peut y retrouver la définition suivante :

Définition 3.1. *(Monotonie selon Lebesgue [42]) Soit Ω un ouvert borné. Une fonctions continue $g : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$ est monotone si pour tout domaine fermé $\Omega' \subseteq \Omega$, g atteint son minimum et son maximum sur $\partial\Omega'$.*

Une difficulté avec les définitions introduites par Lebesgue [42], Mostow [48], Vodopyanov et Goldstein [59] est que les hypothèses posées sur g ne sont vérifiables que si l'on connaît la forme explicite de g. En effet, il n'est pas possible pour l'utilisateur de stipuler si une boîte

noire f possède ses extremums sur les bordures de Ω' et ce pour tout Ω'. Même si l'utilisateur était en mesure d'affirmer que la boîte noire respecte la définition 3.1, cela ne donnerait, à priori, aucune information pertinente supplémentaire traitable par un algorithme de DFO.

Dans le cadre des mathématiques appliquées, un concept bien particulier semble revenir dans différents domaines : les fonctions monotones sur des cônes. Au meilleur de nos connaissances, la structure de ce type de fonctions n'a jamais été exploitée par des algorithmes de DFO. Par contre, quelques ouvrages [54, 57] portent sur l'optimisation non linéaire de fonctions monotones. Rubinov, Tuy et Mays [54] proposent un algorithme d'optimisation globale de cônes coupants inspiré de la méthode des plans coupants. Les auteurs définissent, à l'aide de la notion de points dominants, ce qu'est une fonction croissante sur le quadrant positif $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x_i \geq 0,\ \forall i = 1, 2, \ldots, n\}$.

Définition 3.2. *(Monotonie selon Rubinov, Tuy et Mays [54]) Une fonction $g : \mathbb{R}^n \mapsto \mathbb{R}$ est dite croissante sur $\mathbb{R}^n_+$ si $g(x') \geq g(x)$ lorsque $x' \geq x \geq 0$.*

Tuy [57] utilise la définition 3.2, mais ajoute aussi la notion de croissance sur un hyper rectangle $[a, b] \subset \mathbb{R}^n_+$ si $g(x) \leq g(x')$ lorsque $a \leq x \leq x' \leq b$. Les problèmes étudiés sont tels que la fonction objectif et les contraintes sont croissantes sur le cône $\mathbb{R}^n_+$. Le cas où ces fonctions ne sont que partiellement monotones (selon quelques variables) n'est pas étudié. Que faire dans le cas où $g(x, y) = x^2 - y$ n'est monotone décroissante que dans une seule direction $(0, 1)$? En fait, demander la croissance sur $\mathbb{R}^n_+$ est assez restrictif et il serait préférable de travailler avec une définition plus souple de la monotonie dans le contexte de l'optimisation de boîtes noires.

Les définitions 3.3 et 3.4 sont des généralisations dans $\mathbb{R}^n$ de celles dans $\mathbb{R}^2$ introduites par Van Dyke, Vixie et Asaki [58]. La figure 3.3 permet de visualiser la définition de l'opérateur $\leq_K$ dans $\mathbb{R}^2$.

Définition 3.3. *Soit $K \subseteq \mathbb{R}^n$ un cône convexe et $x, y \in X \subseteq \mathbb{R}^n$. On définie alors $\leq_K$ de la façon suivante :*

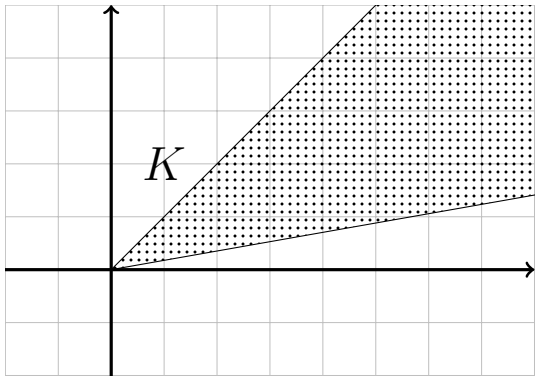
$$x \leq_K y \quad si\ y - x \in K.$$

Définition 3.4. *On dit que $g : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ est **cône-monotone croissante** si pour tout $x \in X$ il existe un cône $K(x)$ convexe tel que*

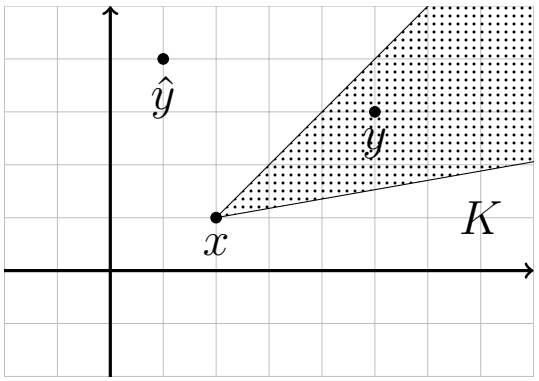
$$g(x) \leq g(y) \quad lorsque\ x \leq_{K(x)} y,\ y \in X.$$

*La fonction est dite K**-monotone croissante** si la fonction est cône-monotone croissante avec un cône fixe K.*

De façon équivalente, nous introduisons les définitions de ***cône-monotone décroissante*** *et* $K-$***monotone décroissante*** *en inversant le sens de l'inégalité* $g(x) \geq g(y)$ *lorsque* $x \leq_{K(x)} y$.



(a) Un cône K centré en $(0, 0)$ dans $\mathbb{R}^2$



(b) Un cône K centré en x tel que $x \leq_K y$, mais $x \not\leq_K \hat{y}$.

Figure 3.3 Représentation graphique de la définition 3.3 de l'opérateur $\leq_K$.

Puisqu'il sera ultérieurement question d'intersection de cônes, il n'est pas spécifié que $K \setminus \{0\} \neq \emptyset$ dans la définition 3.4. Par conséquent, on accepte que toutes les fonctions sont trivialement K-monotones avec $K = \{0\}$. Dans le cas où $K = \mathbb{R}^n_+$, la définition est équivalente à celle de Rubinov et al. (définition 3.2). Dans leur article [18], Borwein, Burke et Lewis définissent la notion de $K-$monotone de façon très similaire. Ils énoncent plusieurs résultats concernant la semi-continuité des fonctions $K-$monotones ainsi que sur la différentiabilité de celles-ci au sens de Hadamard, Gateau et Fréchet [29]. Aussi, ils introduisent des notions telles que les fonctions Lipschiptz-directionnelles. Nous n'explorerons pas ces résultats puisqu'ils sortent largement du cadre de ce travail. Une remarque importante apportée par ces auteurs est qu'une fonction est $K-$monotone croissante si et seulement si elle est $(-K)-$monotone décroissante.

Il y a plusieurs avantages à travailler avec la définition 3.4. Tout d'abord, comme il n'y a aucune condition sur le cône K, la monotonie partielle d'une fonction peut être prise en compte. En effet, K peut avoir un intérieur vide contrairement à la définition 3.2 de Rubinov et al. De plus, cette définition indique que tout comportement de g à l'extérieur d'un domaine d'intérêt X peut être ignoré. Cela s'avère particulièrement utile dans le cas où $g(x, y) = x(y - 2)$ sur $x \in \mathbb{R}_+$ et $y \geq 2$, par exemple. Aussi, comme il le sera présenté plus loin, K est un ensemble de directions de descente, au sens de la définition 2.4, qui peut être exploité directement par MADS. Différentes idées d'implémentation ainsi que leur pseudo-

code seront présentées au chapitre 4. Finalement, aucune condition sur la différentiabilité ou la continuité de g n'est demandée, contrairement à la définition 3.1 de Lebesgue. Cet ensemble de facteurs rend la souplesse de la structure $K-$monotone très intéressante dans le contexte de la DFO.

La proposition suivante s'inspire d'un théorème proposé par Van Dyke, Vixie et Asaki [58]. Il relâche la condition sur f qui devait originairement être continue. De plus, le cône K n'a pas besoin d'être d'intérieur non vide comme il est demandé dans l'article. Le même résultat peut être obtenu avec un cône K qui n'est pas, tout simplement, le vecteur nul.

Proposition 3.1. *Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction semi-continue inférieurement*[1] *(sci). Supposons que f soit K-monotone décroissante avec $K\backslash\{0\} \neq \emptyset$. Pour tout domaine $\Omega \subset \mathbb{R}^n$ borné non vide, f atteint son minimum global sur $\partial\Omega$.*

Démonstration. Si $\text{int}(\Omega) = \emptyset$, alors $\partial\Omega = \Omega$ est un ensemble fermé borné. Puisque f est sci, et que $\partial\Omega$ est compact, le minimum global est atteint sur cet ensemble.
On s'intéresse maintenant au cas où $\text{int}(\Omega) \neq \emptyset$. Puisque $\partial\Omega$ est compact et que f est sci, alors il existe un minimum global sur $\partial\Omega$ et il est atteint. C'est-à-dire qu'il existe $x^*_{\partial\Omega} \in \partial\Omega$ tel que $f(x^*_{\partial\Omega}) \leq f(y)$ pour tout $y \in \partial\Omega$.
Supposons que $f(x^*_{\partial\Omega})$ ne soit pas le minimum global sur Ω. Il existe alors $x^*_\Omega \in \Omega$ à l'intérieur du domaine tel que $f(x^*_\Omega) < f(x^*_{\partial\Omega})$.
Prenons le vecteur non nul $d \in K$, alors $f(x^*_\Omega + td) \leq f(x^*_\Omega)\ \forall t \geq 0$. En particulier pour $\bar{t}$ tel que $x^*_\Omega + \bar{t}d \in \partial\Omega$. Alors

$$f(x^*_{\partial\Omega}) \leq f(x^*_\Omega + \bar{t}d) \leq f(x^*_\Omega) < f(x^*_{\partial\Omega})$$

contradiction. □

La prochaine proposition sera grandement utile pour la suite de ce projet. Elle servira dans la construction de la direction de tendance qui sera ultérieurement présentée. Une représentation graphique de sa preuve est fournie par la figure 3.4.

Proposition 3.2. *Soit K_1 et K_2 deux cônes de $\mathbb{R}^n$. Si $g : \Omega \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ est K_1-monotone et K_2-monotone, alors la fonction g est K-monotone où*

$$K = \text{conv}\{K_1, K_2\}$$

Démonstration. Soit $d \in K$, alors il existe $d_1 \in K_1$, $d_2 \in K_2$ et $\alpha \in [0, 1]$ tel que $d = \alpha d_1 + (1 - \alpha)d_2$.

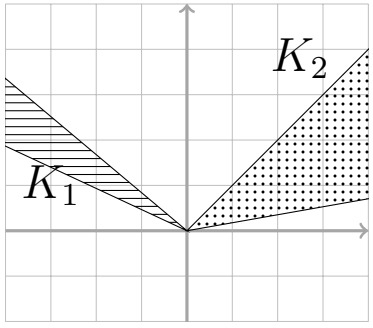
1. La définition peut être retrouvée dans [29].

Soit $y_1 = x + \alpha d_1$ et $y_2 = y_1 + (1-\alpha)d_2$. Ce choix assure la relation d'ordre $x \leq_{K_1} y_1$ et $y_1 \leq_{K_2} y_2$. La définition 3.4, entraîne la suite d'inégalités suivante :

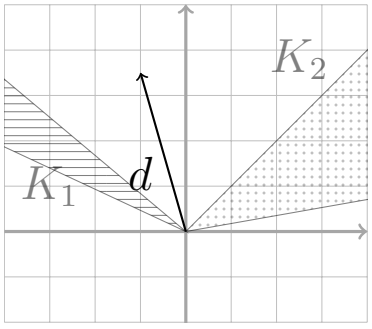
$$\begin{aligned} g(x) &\leq g(x + \alpha d_1) \\ &= g(y_1) \\ &\leq g(y_1 + (1-\alpha)d_2) \\ &= g(x + d) \end{aligned}$$

On a bel et bien que $g(x) \leq g(y)$ lorsque $x \leq_K y$. □

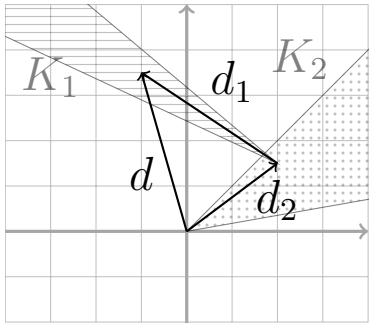
En d'autres termes, lorsque la monotonie d'une fonction est connue sur deux cônes différents, il est possible d'agrandir l'ensemble des directions sur lequel la fonction est monotone à l'aide de l'enveloppe convexe de ces deux cônes. Le corollaire suivant vérifie qu'une direction dans le cône convexe K respecte bel et bien la définition 2.4 d'une direction de descente introduite dans la revue de littérature du chapitre précédent.



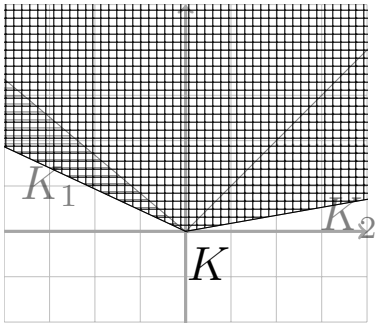
(a) K_1 et K_2 sont deux cônes sur lesquels la fonction g est monotone croissante.



(b) La direction d repose dans l'enveloppe convexe de K_1 et K_2.



(c) Puisque $d_1 \in K_1$ et $d_2 \in K_2$, alors d est nécessairement une direction de montée.



(d) La fonction g est alors K-monotone croissante sur la convexifié des deux cônes de départ.

Figure 3.4 Représentation dans $\mathbb{R}^2$ de la preuve de la proposition 3.2.

Corollaire 3.1. *Soit $g : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ une fonction $K-$monotone décroissante. Si $d \in K \cap R_X(x)$, alors d est une direction de descente $\forall x \in X$.*

Démonstration. Soit $d \in K$ tel que $d \in R_X(x)$, alors la définition 2.4 assure qu'il existe un scalaire $\bar{\alpha} > 0$ tel que $x + \alpha d \in X$ pour tout $\alpha \in]0, \bar{\alpha}]$.
Puisque $d \in K$ alors αd reste dans le cône K. Donc $g(x + \alpha d) \leq g(x)$ et d est une direction de descente avec un choix de $\alpha' = \bar{\alpha}$. □

Même si aucune hypothèse n'est posée sur g dans la définition 3.4, il existe un lien étroit entre le gradient de g et la construction de K lorsque la fonction $g : \mathbb{R}^n \mapsto \mathbb{R}$ est différentiable. Le corollaire suivant en fait la démonstration.

Corollaire 3.2. *Soit $g : \mathbb{R}^n \mapsto \mathbb{R} \in \mathcal{C}^1$, et K le plus grand cône pour lequel g est K-monotone croissante. Alors $d \in K$ si et seulement si $\nabla g^\top(x) d \geq 0, \ \forall x \in \mathbb{R}^n$.*

Démonstration. Supposons que $d \in K$ n'est pas le vecteur nul (cas trivial). Montrons que $d \in K$ implique que $\nabla g(x) \cdot d \geq 0$ pour tout $x \in \mathbb{R}^n$. Par définition de $K-$monotone croissante, on a que $\forall t \geq 0$ et $\forall x \in \mathbb{R}^n$

$$
\begin{aligned}
x \leq_K (x + td) &\Rightarrow g(x + td) \geq g(x) \\
&\Rightarrow g(x + td) - g(x) \geq 0 \\
&\Rightarrow \lim_{t \searrow 0} \frac{g(x + td) - g(x)}{t} \geq 0 \\
&\Rightarrow \lim_{t \to 0} \frac{g(x + td) - g(x)}{t} \geq 0, \quad \text{car } g \in \mathcal{C}^1 \\
&\Rightarrow g'(x; d) \geq 0 \\
&\Rightarrow \nabla g^\top(x) d \geq 0.
\end{aligned}
$$

Montrons par contradiction que $\nabla g(x) \cdot d \geq 0, \ \forall x \in \mathbb{R}^n$ implique que d peut être ajouté au cône K.
Supposons que d ne peut être ajouté à K. Cela veut dire qu'il existe $x \in \mathbb{R}^n$ et $t > 0$ tel que $g(x + td) < g(x)$. Par le théorème des accroissements finis, il existe $\xi \in]x, x + td[$ tel que

$$\nabla g(\xi) \cdot d = g'(\xi; d) = g(x + td) - g(x) < 0.$$

□

Ce corolaire s'avèrera particulièrement utile dans la section 4.1 lorsque la matrice de tendance sera construite directement à partir des équations analytiques du problème à optimiser. Avant

toute chose, une révision de cette matrice de tendance s'impose grâce à tous ces nouveaux concepts théoriques explorés.

3.3 La matrice de tendance

Il est maintenant possible de revoir la définition de la matrice de tendance T sur laquelle reposera la modification de l'algorithme MADS. Celle-ci diffère de l'équation (3.1) puisque qu'elle découle de la théorie de la monotonie sur un cône introduite à la section précédente. Par la suite, plusieurs résultats théoriques ainsi qu'une nouvelle méthode pour calculer d_T seront présentés dans le but de prouver que $-d_T$ est une direction de descente pour la fonction de violation de contraintes $h(x)$. Finalement, nous expliquerons en quoi ce changement de définition de la matrice T vient pallier aux problèmes rencontrés lors de la première approche présentée à la section 3.1.

Définition 3.5. *Soit le problème d'optimisation donné par l'équation* (1.1)*. Les éléments de la* ***matrice de tendance*** T *de taille* $n \times (m+1)$ *sont définis de la façon suivante :*

$$T_{i,j} = \begin{cases} -1 & \textit{si } f_j \textit{ est } K-\textit{monotone décroissante sur } K = \{t\boldsymbol{e}_i \ : \ t \in \mathbb{R}^+\} \\ 1 & \textit{si } f_j \textit{ est } K-\textit{monotone croissante sur } K = \{t\boldsymbol{e}_i \ : \ t \in \mathbb{R}^+\} \\ 0 & \textit{si } f_j \textit{ est constante par rapport à } x_i \\ N/A & \textit{si l'information est inconnue ou qu'il y a absence de monotonie} \end{cases}$$

où $i \in \{1, \ldots, n\}$ *et* $j \in \{0, 1, \ldots, m\}$.

En plus d'utiliser la notion de cônes, cette nouvelle définition de T diffère de l'équation (3.1) par l'ajout d'un élément N/A laissant le symbole 0 représenter un comportement constant. La figure 3.5 de l'exemple 3.2 représente bien le rôle subtil du zéro dans la construction de l'ensemble des directions de montée à partir de la matrice de tendance T. À ce stade, il est important de souligner que les éléments de $T_{i,j}$ peuvent être obtenus soit par une étude analytique approfondie du problème, soit par l'intuition du constructeur de la boîte noire. Nous supposerons pour l'instant que les informations données par l'utilisateur sont vraies.

Les deux éventuels buts d'une telle matrice T sont :

1. Arriver à une solution réalisable plus rapidement grâce aux colonnes $T_1, T_2 \ldots, T_m$.
2. Arriver à une solution optimale plus rapidement grâce à la colonne T_0.

Notons que, selon la définition 3.5 de la matrice de tendance, les éléments de T sont calculés à l'aide de cônes «K» ayant la forme de demi-droites. À l'aide du théorème 3.2, il nous

est possible de trouver $m+1$ cônes convexes plus «larges» sur lesquels les $m+1$ fonctions, associées aux colonnes de la matrice T, sont monotones croissantes. En effet, il suffit de calculer les enveloppes convexes de ces demi-droites, appelées cônes de tendance.

Définition 3.6. *Soit T une matrice de tendance. À partir des colonnes de la matrice T, les $m+1$* ***cônes de tendance*** *sont définis de la façon suivante :*

$$K_{T_j} = \operatorname*{conv}_{i\in\{1,2,\ldots,n\}} \{t \times e_i : t \cdot T_{i,j} \geq 0, t \in \mathbb{R}\} \cup \{0\}, \quad j \in \{0,1,\ldots,m\}$$

avec pour convention que $t \times N/A \not\geq 0$, pour tout $t \in \mathbb{R}^n$.

L'ajout du vecteur nul à l'enveloppe convexe assure que K_{T_j} est non vide dans le cas où T_j est un vecteur colonne uniquement composé de symboles N/A. Cela simplifiera les énoncés qui suivront.

Corollaire 3.3. *Soit le problème de minimisation* (1.1)*. La fonction f_j est K_{T_j}-monotone croissante pour tout $j \in \{0,1,\cdots,m\}$.*

> *Démonstration.* Si $f_j(x)$ est monotone décroissante sur le cône $\{te_i : t \in \mathbb{R}_+\}$, alors elle est monotone croissante sur $\{te_i : t \in \mathbb{R}_-\}$.
> Puisque K_{T_j} est l'enveloppe convexe des demi-droites sur lesquelles f_j est croissante par construction, la proposition 3.2 assure que f_j est K_{T_j}-monotone croissante.
>
> □

Ce dernier corollaire assure que K_{T_j} contient, par construction de l'ensemble, des directions de croissance pour la fonction f_j qui lui est associée. Sachant qu'il est maintenant possible de créer de tels K_{T_j}, ceux-ci serviront au calcul d'une direction de montée valide pour un ensemble de fonctions.

Définition 3.7. *Soit T une matrice de dimension $n \times (m+1)$ et J un ensemble d'indices tel que $J \subseteq \{0,1,\ldots,m\}$. On construit $d_T(J) \in \mathbb{R}^n$ une* ***direction de tendance*** *par respect à l'ensemble J et la matrice T de la façon suivante :*

$$d_T(J) = \sum_{i=1}^{n} s_i \times e_i, \text{ où } s_i = \begin{cases} 1 & \text{si } T_{i,j} \geq 0 \ \forall j \in J \text{ et } \sum_{j\in J} T_{i,j} > 0, \\ -1 & \text{si } T_{i,j} \leq 0 \ \forall j \in J \text{ et } \sum_{j\in J} T_{i,j} < 0, \\ 0 & \text{sinon,} \end{cases}$$

avec pour convention que $N/A \not\geq 0$ et $N/A \not\leq 0$.

Cette dernière convention assure que $s_i = 0$ dès qu'il existe un $j \in J$ tel que $T_{i,j} = N/A$. En d'autres termes, le $i^{ème}$ élément de la direction de tendance est posé comme nul dès que la $i^{ème}$ variable du problème (1.1) n'a pas un effet monotone connu sur toutes les contraintes d'intérêt. Plus particulièrement, dans le prochain chapitre, les contraintes d'intérêt seront celles qui sont violées par la solution actuelle x^k de l'algorithme MADS.

Corollaire 3.4. *Soit T une matrice de tendance, J un ensemble d'indices tel que $J \subseteq \{0, 1, \ldots, m\}$ et $d_T(J) \in \mathbb{R}^n$ la direction de tendance, alors*

$$d_T(J) \in \bigcap_{j \in J} K_{T_j}.$$

Démonstration. Supposons que $d_T(J) \neq 0$. Par définition, on a que $d_T(J) = s_1 e_1 + s_2 e_2 + \cdots + s_n e_n$. Montrons que les vecteurs $s_i e_i$ avec $i \in \{1, \ldots, n\}$ sont dans tous les cônes de tendance K_{T_j} pour tout $j \in J$.

Si $s_i = 1$, alors $T_{i,j} = 0$ ou 1, $\forall j \in J$ par définition de la direction de tendance. Avec un choix de $t = 1$ dans la définition 3.6, on a que $s_i e_i = e_i \in K_{T_j}$, $\forall j \in J$.

Si $s_i = -1$, alors $T_{i,j} = 0$ ou -1, $\forall j \in J$ par définition de la direction de tendance. Avec un choix de $t = -1$ dans la définition 3.6, on a que $s_i e_i = -e_i \in K_{T_j}$, $\forall j \in J$.

Si $s_i = 0$, alors $0 \cdot e_i$ est nécessairement dans l'intersection des cônes de tendance qui contiennent tous l'élément nul par définition.

On a que $d_T(J) = s_1 e_1 + s_2 e_2 + \cdots + s_n e_n$ reste alors dans le cône convexe $\bigcap_{j \in J} K_{T_j}$ □

Puisque, par construction, d_T est dans tous les cônes sur lesquels les fonctions f_j sont K_{T_j}-monotones croissantes pour $j \in J$, cela permet d'assurer que $-d_T$ est une direction de descente sur celles-ci pour tout x dans le domaine de définition X du problème (1.1), dans le cas où $-d_T \in R_X(x)$. Comme il a été souligné précédemment, les colonnes $T_1, T_2, \ldots, T_m$ serviront entre autre à guider MADS dans le but de dénicher une solution réalisable au problème (1.1) plus rapidement. Les résultats théoriques suivants appuient cette démarche, le but étant de démontrer que $-d_T$ est une direction de descente sur h.

Lemme 3.1. *Soit la somme $s(x) = \sum_{j=1}^{m} \lambda_j g_j(x)$ avec $\lambda \in \mathbb{R}^m_+$. Si $g_j : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ sont K_j-monotones croissantes pour tout $j \in J = \{1, 2, \ldots, m\}$, alors $s(x)$ est $K-$monotone croissante sur $K = \cap_{j \in J} K_j$.*

Démonstration. Soit $x \in X$ et $d \in K$ tel que $x + d \in X$, alors $d \in K_i \ \forall i \in I$.

$$\begin{aligned} s(x+d) &= \lambda_1 g_1(x+d) + \lambda_2 g_2(x+d) + \cdots + \lambda_m g_m(x+d) \\ &\geq \lambda_1 g_1(x) + \lambda_2 g_2(x) + \cdots + \lambda_m g_m(x) \\ &= s(x) \end{aligned}$$

□

Lemme 3.2. *Le carré d'une fonction positive $g : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}_+$ $K-$monotone croissante reste $K-$monotone croissante.*

Démonstration. Soit $x \in X$ et $d \in K$ tel que $x + d \in X$,

$$\begin{aligned} g^2(x+d) &= g(x+d)g(x+d) \\ &\geq g(x)g(x) \qquad \text{car } g(x+d) \geq g(x) \geq 0 \\ &= g^2(x) \end{aligned}$$

□

Lemme 3.3. *Soit $g(x) = \max\{g_1(x), g_2(x)\}$ où $g_1(x) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ et $g_2(x) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ sont respectivement monotones sur les cônes K_1 et K_2. Alors $g(x)$ est monotone sur le cône $K = K_1 \cap K_2$.*

Démonstration. Soit $x \in X$ et $d \in K = K_1 \cap K_2$ tel que $x + d \in X$,

$$\begin{aligned} g(x+d) &= \max\{g_1(x+d), g_2(x+d)\} \\ &\geq \max\{g_1(x), g_2(x)\} \\ &= g(x) \end{aligned}$$

□

Proposition 3.3. *Soit T la matrice de tendance de dimension $n \times (m+1)$ associée au problème d'optimisation* (1.1)*. Soit $x \in X$ et la direction de tendance $d_T(J(x)) \in R_X(x)$ avec l'ensemble d'indices $J(x) = \{j \in \{1, 2, \ldots, m\} : f_j(x) \geq 0\}$. Alors $-d_T(J(x))$ est une direction de descente sur $h(x)$ si les fonctions $f_j(x)$ sont semi-continues supérieurement*[2] *(scs) en x pour tout $j \in J^C(x) = \{j \in \{1, 2, \ldots, m\} : f_j(x) < 0\}$.*

Démonstration. Soit $x \in X$, $d \in K = \cap_{j \in J}(-K_{T_j})$ une direction réalisable au sens de la définition 2.3, alors il existe $\bar{\alpha} > 0$ tel que $x + \alpha d \in X$ pour tout

2. La définition est donnée par Delfour dans [29]

$0 < \alpha < \bar{\alpha}$. Écrivons la fonction $h(x + \alpha d) = \sum_{j=1}^{m} \max\{f_j(x + \alpha d), 0\}^2$ sous la forme $h(x + \alpha d) = S_1(x + \alpha d) + S_2(x + \alpha d)$ où

$$S_1(x) = \sum_{j \in J(x)} \max\{f_j(x), 0\}^2,$$
$$S_2(x) = \sum_{j \in J^C(x)} \max\{f_j(x), 0\}^2.$$

La première partie de la preuve consiste à montrer que $S_1(x + \alpha d) \leq S_1(x)$. Ensuite, il sera montré qu'il existe un $\hat{\alpha} > 0$ assez petit pour que $S_2(x + \alpha d) = S_2(x) = 0$ pour tout $\alpha \in]0, \hat{\alpha}]$.

Montrons que $S_1(x + \alpha d) \leq S_1(x)$. Comme la fonction $g(x) = 0$ est monotone décroissante sur le cône convexe $\mathbb{R}^n$, le lemme 3.3 assure que la fonction $\max\{f_j(x), 0\}$ est monotone décroissante sur le cône

$$-K_{T_j} \cap \mathbb{R}^n = -K_{T_j}.$$

Comme la fonction $\max\{f_j(x), 0\}$ avec $j \in J$ est toujours positive, son carré est alors aussi $-K_{T_j}$-monotone décroissante selon le lemme 3.2. Finalement, comme $S_1(x)$ est une somme de fonctions, elle est K-monotone décroissante sur le cône

$$K = \cap_{j \in J}(-K_{T_j})$$

par le lemme 3.1. Au corollaire 3.4, il a été montré que la direction de tendance $d_T(J) \in \cap_{j \in J} K_{T_j}$ par construction, ce qui implique que $-d_T(J) \in \cap_{j \in J}(-K_{T_j})$. Avec le choix particulier de $d = -d_T(J)$, on a bel et bien une direction de descente sur S_1 en x par le corollaire 3.1 car $-d_T(J)$ est une direction réalisable.

Montrons qu'il existe $\hat{\alpha} > 0$ tel que $S_2(x + \alpha d) = 0$ pour tout $\alpha \in]0, \hat{\alpha}]$. Comme les fonctions $f_j(x)$ pour $j \in J^C(x)$ sont semi-continues supérieurement en x, alors il existe des voisinages $V_j(x)$ tels que $0 > f_j(y),\ \forall y \in V_j(x)$ et $\forall j \in J^C(x)$. Comme tous les voisinages sont des ouverts non vides, posons

$$y \in V = \cap_{j \in J^C(x)} V_j(x).$$

Dans le cas particulier où $y = x + \bar{\alpha}(-d_T(J))$ pour un $\bar{\alpha} > 0$, on a bel et bien que $S_2(x - \alpha d_T(J)) = 0$ pour tout $\alpha \in]0, \bar{\alpha}]$.

En conclusion $h(x - \alpha d_T(J)) \leq h(x)$ lorsque $0 < \alpha < \alpha' = \min\{\hat{\alpha}, \bar{\alpha}\}$. L'inverse de la direction de tendance est bel et bien une direction de descente. □

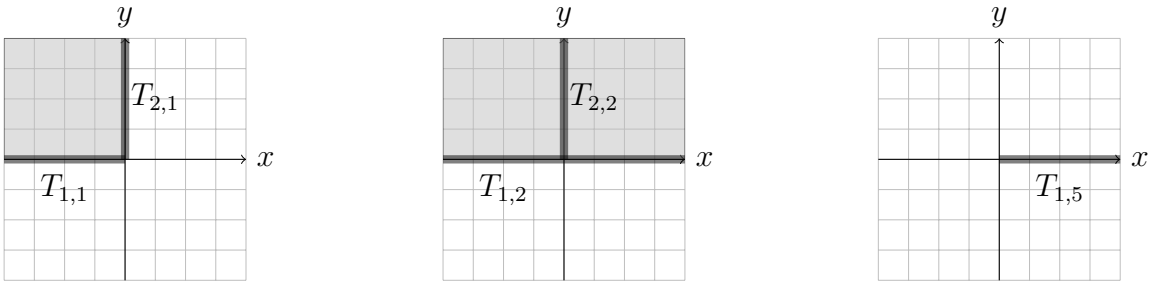
Il est nécessaire que les fonctions f_j pour $j \in J^C(x)$ soient scs en x afin de prouver que l'inverse de la direction de tendance est bel et bien une direction de descente. En effet, comme aucune hypothèse sur la continuité des fonctions n'est posée, la condition scs assure qu'il n'y aura pas de saut brusque de la fonction $h(x)$ pour un petit déplacement dans le sens de $-d_T(J)$. En fait, il faut que les valeurs des contraintes qui sont respectées en x restent négatives pour un petit déplacement. Ce n'est pas toujours le cas si les fonctions ne sont pas scs en x.

Il est donc relativement facile de construire une direction de descente sur la fonction $h(x)$. Notons que si $\{0\}$ est ajouté à l'ensemble J, alors $-d_T$ devient en plus une direction de descente sur la fonction objectif. Cette information devient intéressante lorsque $d_T \neq 0$.

Reprenons l'exemple 3.1 introduit au début de ce chapitre. Nous y appliquerons la nouvelle définition 3.5. La figure 3.5 présentera différentes formes possibles de cônes de tendance construits à partir de la matrice de tendance obtenue.

Exemple 3.2. *La matrice T contient les informations concernant le sens des demie-droites coordonnées sur lesquelles la fonction objectif et les contraintes sont monotones croissantes. Grâce à la proposition 3.2, il nous est possible de tracer le cône K qui admet la monotonie des fonctions grâce à l'enveloppe convexe des demi-droites. Il est important de noter qu'une valeur $T_{i,j}$ nulle dans la matrice T indique que la fonction associée à la colonne j de T est constante par rapport à la variable x_i. La figure 3.5 représente quelques cônes que l'on peut tracer grâce à T :*

$$T = \begin{pmatrix} 1 & -1 & 0 & -1 & 1 & 1 \\ 0 & 1 & 1 & -1 & -1 & N/A \end{pmatrix}$$



(a) K_{T_1} (b) K_{T_2} (c) K_{T_5}

Figure 3.5 Différents types de cônes selon la matrice de tendance sur lesquels (a)f_1, (b)f_2 et (c)f_5 sont monotones croissantes

Dans le cas où $x^0 = (4, 3)^\top$, la direction de tendance $d_T(J(x^0))$ se trouve dans l'intersection

des cônes K_{T_2} et K_{T_5}. Par la définition 3.7, on obtient $d_T(J(x^0)) = (1,0)^\top$. Cette situation est représentée par la figure 3.6.

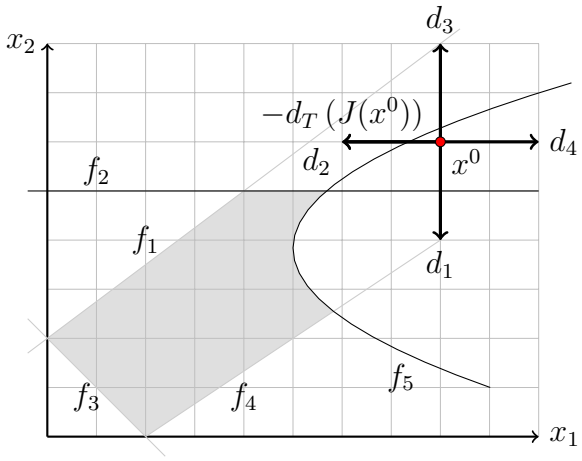


Figure 3.6 Étape de sonde locale autour de $x^0 = (4,3)^\top$

La proposition 3.3 assure donc que la direction de tendance construite selon la définition 3.7 est une direction de montée sur h. Rappelons que la première méthode introduite dans la section 3.1, trop naïve et simple, n'avait pas ce luxe. Cette nouvelle façon de traiter la monotonie vient pallier aux autres problèmes rencontrés dans la section 3.1. En effet, la méthode ne dépend plus d'une pondération des éléments de T. De plus, si le comportement de la fonction objectif f_0 doit aussi être pris en compte, il est suffisant d'ajouter l'élément $\{0\}$ dans l'ensemble $J(x)$. Il n'est donc plus nécessaire d'introduire un poids $\xi > 0$ de taille subjective.

Maintenant que la base théorique est dressée, le prochain chapitre s'attardera à la partie pratique du projet. Des pseudo-codes présenteront les différentes façons d'intégrer les informations contenues dans la matrice de tendance dans l'algorithme MADS.

CHAPITRE 4 DESCRIPTION DES ALGORITHMES, PROBLÈMES TESTS ET RÉSULTATS

Ce chapitre propose deux méthodes bien différentes pour parvenir à l'exploitation des informations de la matrice de tendance T par un algorithme de recherche par motifs. Une première technique, qui a déjà été suggérée dans les exemple 3.1 et 3.2 du chapitre précédent, consiste à ordonner l'ensemble des directions candidates de la sonde locale de MADS par rapport à l'inverse de la direction de tendance. Cette approche a été élaborée alors que seule l'heuristique de la section 3.1 était disponible. Une seconde méthode plus agressive serait d'évaluer directement la boîte grise dans la direction opposée à d_T grâce à une recherche linéaire implantée dans l'étape de recherche globale. La section 4.1 présente en détails ces deux méthodes ainsi que les pseudo-codes. Ensuite, trois différentes techniques pour trouver le contenu de la matrice T seront explorées. La section 4.2 expose les problèmes tests académiques et industriel sur lesquels seront mis à l'épreuve toutes les variantes algorithmiques d'intérêt de NOMAD. De plus, les profils de performance et de données seront fournis. Une discussion sur les résultats obtenus est présentée pour chacun des problèmes tests. Une synthèse plus globale contenant des remarques, des recommandations et des idées de travaux futurs est réservée pour le chapitre 5.

Il est à noter que, dans le cadre de ce travail, une attention particulière est portée à l'optimisation de la fonction h quantifiant la violation des contraintes. En effet, l'activation de la barrière progressive rendait l'analyse sur l'impact de T difficile lorsque NOMAD optimisait sur un itéré courant primaire réalisable. D'autant plus que, dans deux problèmes tests sur trois, aucune information sur la monotonie de la fonction objectif n'était disponible. D'autres raisons, qui seront évoquées dans la discussion de la conclusion, nous laissent croire que l'impact de la matrice de tendance est négligeable lors de l'amélioration de la fonction objectif. Les profils de performance et de données sur l'amélioration de la fonction objectif ont donc été ajoutés à titre indicatif seulement, car la matrice de tendance pouvait tout de même avoir un impact autour d'un itéré courant secondaire non réalisable. Cependant, seule l'amélioration de h sera sujet de discussions.

4.1 Différentes modifications de MADS

4.1.1 Ordonnancement de la sonde locale

Une première forme d'intégration des informations de la matrice de tendance est par l'ordonnancement des directions de recherche dans l'étape de sonde locale. Puisque NOMAD profite une stratégie opportuniste, il est attendu que la matrice de tendance permette de sauver quelques évaluations de la boîte noire. La modification de l'étape de sonde locale est présentée par le pseudo-code 4. Puisque NOMAD utilise la méthode de la barrière progressive, il est possible qu'il y ait deux itérés courants : un point réalisable x^k_{fea} et un point non réalisable x^k_{inf}. Dans un tel cas, les ensembles $P^k(x^k_{inf}) = \{x^k_{inf} \pm \delta^k d : d \in D^k_\Delta\}$ et $P^k(x^k_{fea}) = \{x^k_{fea} \pm \delta^k d : d \in \bar{D}^k_\Delta\}$ ne sont pas construits à partir du même ensemble de directions D^k_Δ. Si x^k_{inf} est déclaré comme itéré courant primaire, alors D^k_Δ est une base positive de $\mathbb{R}^n$ et $\bar{D}^k_\Delta$ est un ensemble de cardinalité plus petite et vice versa. Il faut donc calculer deux directions de tendance liées aux itérés courants afin de trier indépendamment D^k_Δ et $\bar{D}^k_\Delta$ en priorisant les vecteurs qui ont le plus petit angle avec les inverses des directions de tendance associées au point réalisable et non réalisable respectivement.

Méthodologie

À la dernière étape de la sonde locale de l'algorithme 4, il est demandé d'évaluer tous les candidats de $P^k(x^k_{inf}) \cup P^k(x^k_{fea})$. En pratique, lorsque les deux ensembles ne sont pas vides, ceux-ci sont concaténés à la manière d'un jeu de carte afin de tirer profit au maximum de la stratégie opportuniste. C'est-à-dire que les ensembles sont coupés en deux et la première tranche de l'ensemble P^k associé à l'itéré courant primaire est priorisée. Par la suite, la première tranche du second ensemble est introduite et ainsi de suite.

NOMAD utilise une mise à l'échelle dynamique des variables selon la méthode introduite en 2014 par Audet, Le Digabel et Tribes [14] lors de l'adaptation de son treillis M^k. Le scalaire Δ^k est alors transformé en vecteur $\Delta^k \in \mathbb{R}^n$ où chaque élément est mis à jour indépendamment. Cette échelle dynamique permet de s'attaquer, par exemple, à des problèmes où une variable est bornée entre 0 et 1000 alors qu'une deuxième se situe entre 0 et 0.1. Il faut donc porter une attention particulière au calcul de l'angle entre la direction de tendance et une direction de recherche proposée par MADS à l'étape de sonde locale. En effet, celles-ci se trouvent sur deux espaces possiblement «étirés» différemment. Avant d'effectuer le produit scalaire entre ces deux directions afin de trouver l'angle, une remise à l'échelle des directions de la sonde locale est effectuée grâce à une division terme à terme avec les éléments du vecteur Δ^k. Cette étape d'échelonnement permet de comparer les directions candidates sur un espace semblable

à celui de la direction de tendance.

Cet algorithme a été créé alors que seule la méthode heuristique de la section 3.1 pour l'exploitation de la matrice de tendance était connue. Par la suite, une seconde approche a été présentée avec des résultats théoriques beaucoup plus forts concernant la construction de la direction de tendance. Présentement, l'ordonnancement de l'ensemble générateur positif D_{Δ}^{k} par $-d_T$ ne permet pas de tirer profit au maximum de la proposition 3.3 selon laquelle $-d_T$ est une direction de descente sur h. La prochaine section s'adresse à ce problème grâce à une recherche linéaire dans le sens opposé à la direction de tendance.

Algorithme 4 : Modification de la sonde locale de MADS

Soit $f : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}^{(m+1)}$ avec f_0 la fonction objectif et $f_j, j = 1, 2, \ldots, m$ les contraintes. Soit x^0 un point de départ réalisable ou non.

Étapes 0, 1 et 2

Voir le pseudo-code 3 de MADS

T la matrice de tendance de dimension $n \times (m+1)$

3. Sonde locale

x^k_{inf} l'itéré courant non réalisable (si existe)

x^k_{fea} l'itéré courant réalisable (si existe)

si x^k_{inf} *existe* **alors**

Poser $J(x^k_{inf}) = \{j \in \{1, \ldots, m\} : f_j(x^k_{inf}) > -\epsilon\}$ l'ensemble d'indice des contraintes actives

Calculer $d_T\left(J(x^k_{inf})\right)$ selon la définition 3.7

Ordonner D^k_Δ en ordre croissant de l'angle avec $-d_T\left(J(x^k_{inf})\right)$

Créer $P^k(x^k_{inf}) = \{x^k_{inf} \pm \delta^k d : d \in D^k_\Delta\}$, l'ensemble ordonné des candidats à évaluer

sinon

$P^k(x^k_{inf}) = \emptyset$

si x^k_{fea} *existe* **alors**

Poser $J(x^k_{fea}) = \{j \in \{1, \ldots, m\} : f_j(x^k_{fea}) > -\epsilon\} \cup \{0\}$ l'ensemble d'indice des contraintes actives et de la fonction objectif

Calculer $d_T\left(J(x^k_{fea})\right)$ selon la définition 3.7

Ordonner $\bar{D}^k_\Delta$ en ordre croissant de l'angle avec $-d_T\left(J(x^k_{fea})\right)$

Créer $P^k(x^k_{fea}) = \{x^k_{fea} \pm \delta^k d : d \in \bar{D}^k_\Delta\}$, l'ensemble ordonné des candidats à évaluer

sinon

$P^k(x^k_{fea}) = \emptyset$

Utiliser la barrière progressive avec une stratégie opportuniste sur l'ensemble des candidats $P^k(x^k_{inf}) \cup P^k(x^k_{fea})$.

4. Arrêt

Voir le pseudo-code 3 de MADS

4.1.2 Recherche linéaire dans l'étape de recherche globale

Une seconde méthode qui sera testée est l'intégration d'une recherche linéaire (LS) très primitive et peu gloutonne à l'étape de la recherche globale de MADS. Selon la proposition 3.3, la construction de la direction de tendance, dans le cas où les fonctions sont semi-continues supérieurement, assure que $-d_T$ est une direction de descente sur la fonction h. Si l'indice 0 est ajouté à $J(x)$, alors $-d_T(J(x))$ est aussi une direction de descente pour la fonction objectif. Ce n'est pas le cas de l'algorithme 4 présenté précédemment qui ne garantit pas que les candidats évalués auront une amélioration de h ou f_0. Une recherche linéaire dans la seconde étape de MADS assurerait que le déplacement effectué est bel et bien dans le même sens qu'une direction de descente. Le pseudo-code est donné par l'algorithme 5.

Méthodologie

En pratique, la recherche linéaire a été implémentée de la façon suivante. Après avoir calculé la direction de tendance associée à l'un où l'autre des itérés courants, la longueur du pas est déterminée par la moyenne entre la taille du treillis et celle du treillis. Un second point, dont la longueur du pas est deux fois la taille du treillis, est aussi ajouté à la recherche linéaire. Donc $S^k(x^k) = \left\{x^k - \frac{(\delta^k+\Delta^k)}{2} d_T\left(J(x^k)\right)\right\} \cup \left\{x^k - 2\delta^k d_T\left(J(x^k)\right)\right\}$ ne contient que deux éléments. Par la suite, ces deux éléments sont projetés sur le treillis afin de respecter les critères de convergence de MADS [8].

Algorithme 5 : Recherche linéaire dans la recherche globale de MADS

Soit $f : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}^{(m+1)}$ avec f_0 la fonction objectif et $f_j, j = 1, 2, \ldots, m$ les contraintes. Soit x^0 un point de départ réalisable ou non.

Étapes 0 et 1

Voir le pseudo-code 3 de MADS

T la matrice de tendance de dimension $n \times (m + 1)$

2. Recherche globale

x^k_{inf} l'itéré courant non réalisable (s'il existe)

x^k_{fea} l'itéré courant réalisable (s'il existe)

si x^k_{inf} *existe* **alors**

Poser $J(x^k_{inf}) = \{j \in \{1, \ldots, m\} : f_j(x^k_{inf}) > -\epsilon\}$ l'ensemble d'indice des contraintes actives

Calculer $d_T\left(J(x^k_{inf})\right)$ selon la définition 3.7

Créer $S^k(x^k_{inf})$, l'ensemble des candidats générés avec LS dans la direction $-d_T\left(J(x^k_{inf})\right)$ à partir de x^k_{inf}

sinon

$S^k(x^k_{inf}) = \emptyset$

si x^k_{fea} *existe* **alors**

Poser $J(x^k_{fea}) = \{j \in \{1, 2, \ldots, m\} : f_j(x^k_{fea}) > -\epsilon\}$ l'ensemble d'indice des contraintes actives

Calculer $d_T\left(J(x^k_{fea})\right)$ selon la définition 3.7

Créer $S^k(x^k_{fea})$, l'ensemble des candidats générés avec LS dans la direction $-d_T\left(J(x^k_{fea})\right)$ à partir de x^k_{fea}

sinon

$S^k(x^k_{fea}) = \emptyset$

Utiliser la barrière progressive avec une stratégie opportuniste sur l'ensemble des candidats $S^k(x^k_{inf}) \cup S^k(x^k_{fea})$.

Étapes 3 et 4

Voir le pseudo-code 3 de MADS

4.1.3 Description des algorithmes testés

Les problèmes tests ainsi que les résultats sont présentés dans la prochaine section. Plusieurs abréviations seront utilisées afin d'alléger les légendes des profils de performance ainsi que des profils de données. La liste suivante donne la signification des mots-clefs qui seront utilisés.

- N+1 : La base positive D_{Δ}^{k} retrouvée à l'étape de sonde locale est composée de $n+1$ directions.
- 2N : L'ensemble générateur positif D_{Δ}^{k} retrouvée à l'étape de sonde locale est composée de $2n$ directions.
- PB : La méthode de la barrière progressive est activée.
- T : La matrice de tendance T est utilisée pour l'ordonnancement des candidats à l'étape de sonde locale selon l'algorithme 4.
- QUAD : Les modèles quadratiques sont activés dans l'étape de recherche globale, afin de trouver les candidats les plus prometteurs, et dans l'étape de sonde locale, afin de créer et d'ordonner les directions de recherche. La barrière progressive PB est aussi activée.
- LS : La matrice de tendance T est utilisée pour une recherche linéaire selon l'algorithme 5.

Toutes les résolutions de problèmes ont été menées par le logiciel d'optimisation NOMAD version 3.7.3 . Les options N+1, 2N, PB et QUAD étaient déjà implémentées dans le programme. Seules les nouvelles méthodes (T et LS) présentées dans cette section ont été ajoutées dans le langage C++ à des fins expérimentales.

Puisque la construction de la matrice de tendance T demande une bonne connaissance du problème d'optimisation ainsi qu'une implémentation spécialisée, les tests numériques n'ont pas pu être menés sur un grand nombre de problèmes. La construction de T aurait pu être automatisée à l'aide d'une méthode par échantillonnage telle que présentée dans la prochaine section. Cependant, dans une large banque de problèmes, comme celle de Hock-Schittkowski [38], seule une minorité de problèmes auraient pu engendrer une matrice de tendance intéressante pour la construction de directions de tendance non nulles.

Plusieurs points de départs ont étés générés à l'aide d'un hypercube latin dans l'ensemble de définition X des différents problèmes. Selon le temps de calcul de ces problèmes, le nombre de points de départ peut différer.

4.1.4 Trois façons de déterminer le contenu de la matrice de tendance

Selon la nature du problème, il y a plusieurs façons d'établir le contenu de la matrice de tendance associée à celui-ci. Dans ce projet, trois méthodes particulières seront explorées : l'approche analytique et par échantillonnage ainsi que l'intuition de l'utilisateur. L'exploration de ces techniques est motivée par la forme des trois problèmes tests qui seront présentés dans la section suivante. Les différents calculs qui ont été menés méritent des explications approfondies et font donc l'objet de la présente section.

La méthode analytique met à profit le corollaire 3.2 introduit précédemment selon lequel une direction d peut être introduite dans le cône de monotonie K si, pour tout point x, le gradient de la fonction $f(x) \in \mathcal{C}^1$ et la direction d ont un produit scalaire positif. Cela s'avère particulièrement utile lorsque les éléments du gradient conservent le même signe en tout point sur le domaine d'intérêt X du problème (1.1). Si, par exemple, le premier élément du gradient de la fonction objectif $\nabla f_0(x)$ est toujours positif, alors on posera $T_{1,0} = +1$. Dans le cas où un changement de signe est observé, alors la matrice de tendance contiendra l'élément N/A. Le problème test `G2` adopte cette approche, car les contraintes sont simples et explicitement connues.

Une seconde méthode utilise un large ensemble de points générés aléatoirement dans le domaine X à l'aide d'un hypercube latin. Quelques pistes concernant la détection d'une monotonie par échantillonnage ont été étudiées. Cependant, ce sujet est rapidement apparu comme complexe et pourrait faire l'objet d'un projet en lui-même. Parmi les méthodes explorées, c'est la plus primitive qui a été retenue pour la facilité de sa programmation. De plus, la qualité des informations fournies était suffisante dans le cadre de ce travail. L'idée est de mettre à profit la prémisse de base selon laquelle : si une fonction f est monotone croissante dans une direction d alors, pour tout point x, on observe que $f(x) \leq f(x + d)$. En fait, elle est une condition nécessaire, mais non suffisante, à l'existence d'une monotonie. Le code `MATLAB` qui a été créé fonctionne de la façon suivante. Une centaine de points de $\mathbb{R}^n$ sont générés par un hypercube latin sur le domaine borné X. Ensuite, à partir de chacun de ces points, n points supplémentaires ont été créés par un déplacement $d_i = \delta_i e_i$ dans le sens des coordonnées avec $i = 1, ..., n$ et $\delta_i > 0$. Finalement, pour chaque d_i, le test suivant est effectué. Si pour tout point x de départ généré par l'hypercube latin on a que $f(x) \leq f(x + d_i)$, alors on suppose que la fonction est monotone croissante. S'il existe un seul point de l'hypercube latin tel que $f(x) > f(x + d_i)$, alors la fonction ne peut être considérée comme monotone croissante dans la direction d_i. Cette méthode est primitive, car les n longueurs de pas δ_i sont restés fixes lors des tests numériques. Un raffinement possible à cette dernière serait de créer des pas de longueur aléatoire $\delta_i > 0$ pour autant que $x + d_i$ reste dans X. Le problème `MDO` fera appel à

cette tactique pour la construction de la matrice de tendance. L'approche est similaire pour détecter un effet monotone décroissant.

Finalement, la dernière méthode pour remplir la matrice de tendance est certainement la plus dangereuse, car elle consiste à donner une confiance aveugle à l'intuition de l'utilisateur de la boîte noire. Cela sera le cas pour le problème `Kemano` qui repose sur des contraintes ayant un sens physique. Nous montrerons qu'il est avantageux d'effectuer une vérification des informations fournies à l'aide de l'échantillonnage présenté plus haut.

4.2 Les problèmes tests

4.2.1 `G2`

Le problème `G2` suivant provient d'une liste de problèmes tests proposée par Michalewicz et Shoenauer [46] en 1996. Il est défini de la façon suivante :

$$\max_{x\in\mathbb{R}^n} \; f_0(x) = \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2\prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right|$$

sous les contraintes

$$f_1(x) = -\prod_{i=1}^n x_i + 0.75 \leq 0$$

$$f_2(x) = \sum_{i=1}^n x_i - 7.5n \leq 0$$

et les bornes $0 \leq x \leq 10$. Afin de faciliter la création de points de départs non réalisables, nous avons repoussé la borne supérieure à 25.

Ce problème est considéré comme difficile par Michalewicz et Schoenauer [46] ainsi que par Audet, Dennis et Le Digabel [12], car la solution optimale se trouve près de l'origine, où f_0 est mal définie et où le produit contenu dans f_1 s'annule dès qu'une variable est fixée à 0. Il est donc difficile pour un algorithme par recherche directe de sortir des axes coordonnés en bougeant, dans le sens positif, toutes les variables nulles en même temps afin de trouver une solution réalisable. Au fil des tests numériques, il a été remarqué qu'il est difficile pour NOMAD d'éviter ce piège lorsque la barrière progressive est activée et que la dimension du problème est grande. En effet, coller les variables aux bornes inférieures permet de rapidement respecter la contrainte f_2 (et donc minimiser $h(x)$) tout en maintenant une «bonne valeur» de la fonction objectif. Le logiciel se trouve alors souvent piégé avec un produit nul dans la contrainte $f_1(x)$ dont il est difficile de se départir.

Dans l'optique de ce travail, le choix du problème `G2` comporte de nombreux avantages. Tout

d'abord, la dimension du problème est variable et nous permet donc d'étudier la performance de nos algorithmes en fonction de la taille du problème. De plus, puisque la forme analytique du problème est à notre portée, nous pouvons déduire avec justesse la valeur des éléments qui seront posés dans T et ne dépendront pas que d'une intuition ou d'un échantillon. Donc, même si `G2` n'est pas une boite grise au sens de la définition 2.2, le problème reste intéressant dans le cadre de ce travail, car les informations contenues dans T seront exactes et assureront une bonne fiabilité lors de l'analyse des résultats obtenus.

Construction de la matrice T

La construction de la matrice de tendance repose sur la méthode analytique présentée dans la section précédente. Puisque la fonction objectif contient des fonctions périodiques, il n'est pas possible de déduire un effet monotone parmi les n variables. La première colonne T_0 est alors composée que de N/A. La première contrainte f_1 est un produit de variable positives. Une augmentation de n'importe quelle variable entraine une diminution de la contrainte. Il est alors possible de poser $T_1 = (-1, \ldots, -1)^\top$. L'idée est la même avec la somme retrouvée dans l'expression de la seconde contrainte $f_2(x)$, ce qui implique que $T_2 = (1, \ldots, 1)^\top$. La matrice de tendance entière est

$$T = \begin{pmatrix} N/A & -1 & 1 \\ \vdots & \vdots & \vdots \\ N/A & -1 & 1 \end{pmatrix}_{n \times 3}.$$

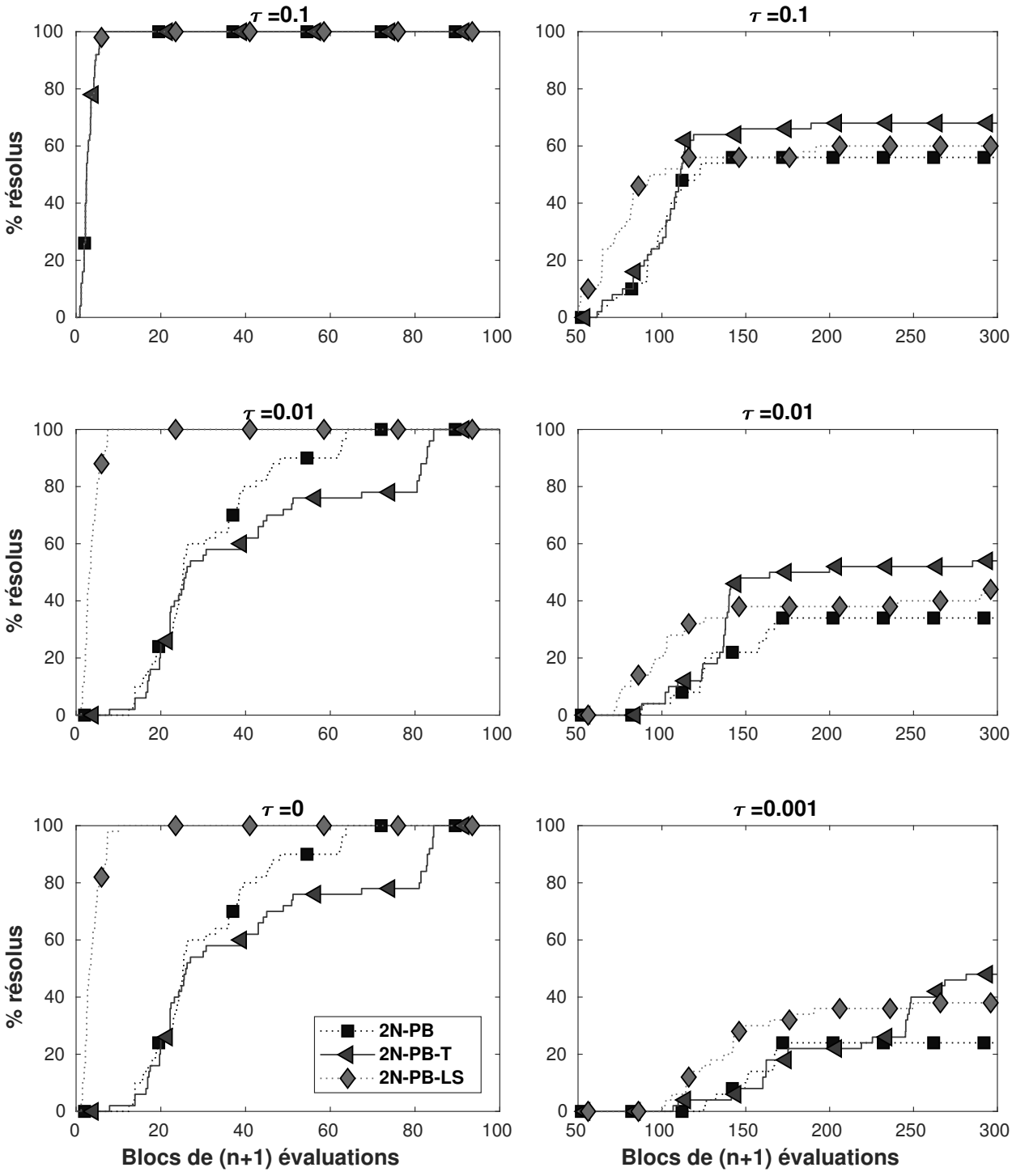
Résultats

Les tests numériques ont été effectués sur deux dimensions différentes du problème, soient $n = 5$ et 10. Pour chaque dimension, les profils de performance et de données ont été tracés sur l'amélioration des contraintes et de la fonction objectif pour un total de quatre figures : 4.1, 4.2, 4.3 et 4.4. Celles-ci ont toutes la même structure. Chacune d'elle contient 6 graphiques. Les trois de la colonne de gauche sont ou bien des profils de performance ou de données associés à l'amélioration de $h(x)$ quantifiant la violation des contraintes. Tel qu'il a été précisé dans la revue de littérature, un $\tau = 0$ permet d'observer le nombre exact d'algorithmes qui ont trouvé une solution réalisable. Les trois graphiques de la colonne de droite concernent l'amélioration de la fonction objectif. Les tests ont été lancés sur 50 points de départ non réalisables générés de la façon suivante. Un hypercube latin a été lancé sur les bornes $15 \leq x \leq 25$ afin que plusieurs points ne respectent pas la contrainte $f_2(x)$. Ensuite, un nombre de $\lfloor 0.75n \rfloor$ éléments ont été posés à 0 de façon aléatoire pour chacun des candidats. Cela

garantit que la liste de points de départ ne respectent pas la contrainte $f_1(x)$.

Les figures 4.1a, 4.2a, 4.3a et 4.4a montrent une dominance importante de la recherche linéaire 2N-PB-LS lors de l'amélioration de la fonction h quantifiant la violation des contraintes, et ce, pour les deux dimensions testées. Cependant, avec $n = 5$, c'est la méthode ordonnancée 2N-PB-T à l'aide de la matrice de tendance qui semble offrir une meilleure performance sur l'amélioration de la fonction objectif selon les figures 4.1b et 4.2b, malgré sa vitesse plus lente à trouver une solution réalisable. Les algorithmes 2N-PB et 2N-PB-T n'ont pas trouvé suffisamment de solutions réalisables en $n = 10$ pour se distinguer dans les graphiques 4.3a et 4.4a par rapport à 2N-PB-LS.

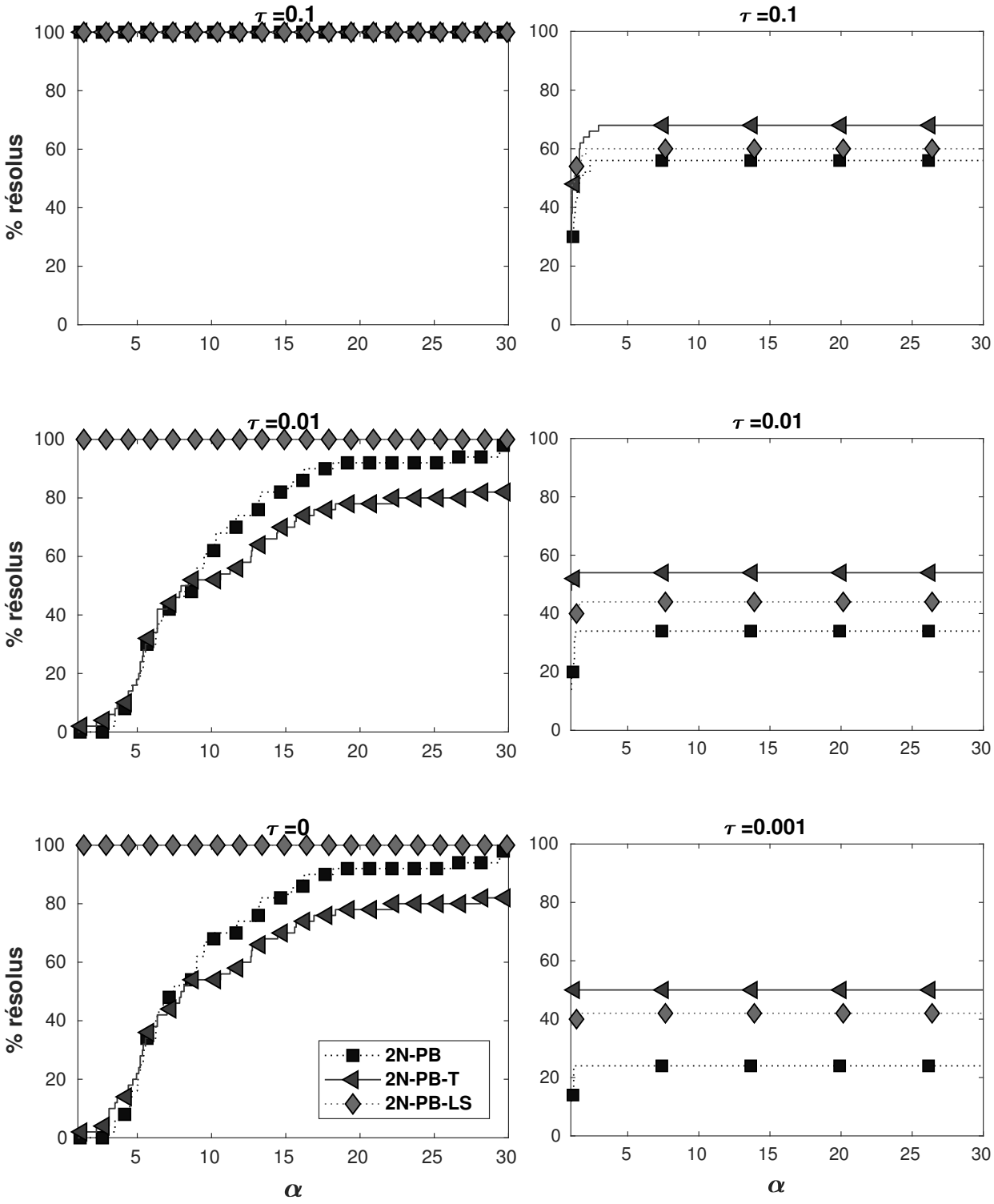
Le problème G2 représente certainement un cas idéal qui permet de mettre en valeur les atouts de la méthode proposée dans ce travail, car la structure du problème est disponible et est exploitée directement. Cependant, nous avons tout de même choisi de le présenter, car il met en lumière une grande amélioration de MADS, sur l'amélioration de h, pour le peu d'efforts consacrés à la construction de sa matrice de tendance T.



(a) Violation des contraintes

(b) Fonction objectif

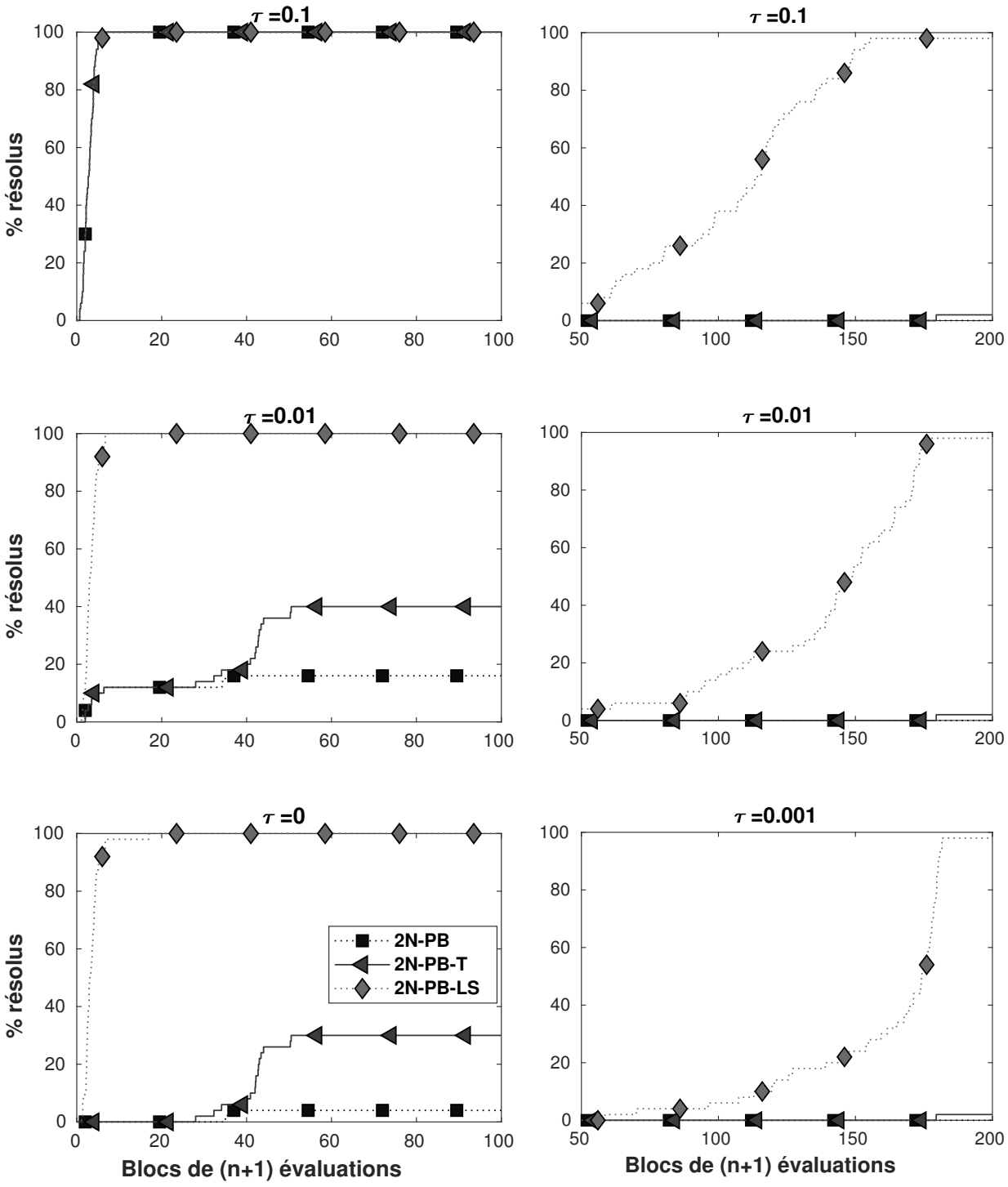
Figure 4.1 Profils de données sur `G2` avec $n = 5$



(a) Violation des contraintes

(b) Fonction objectif

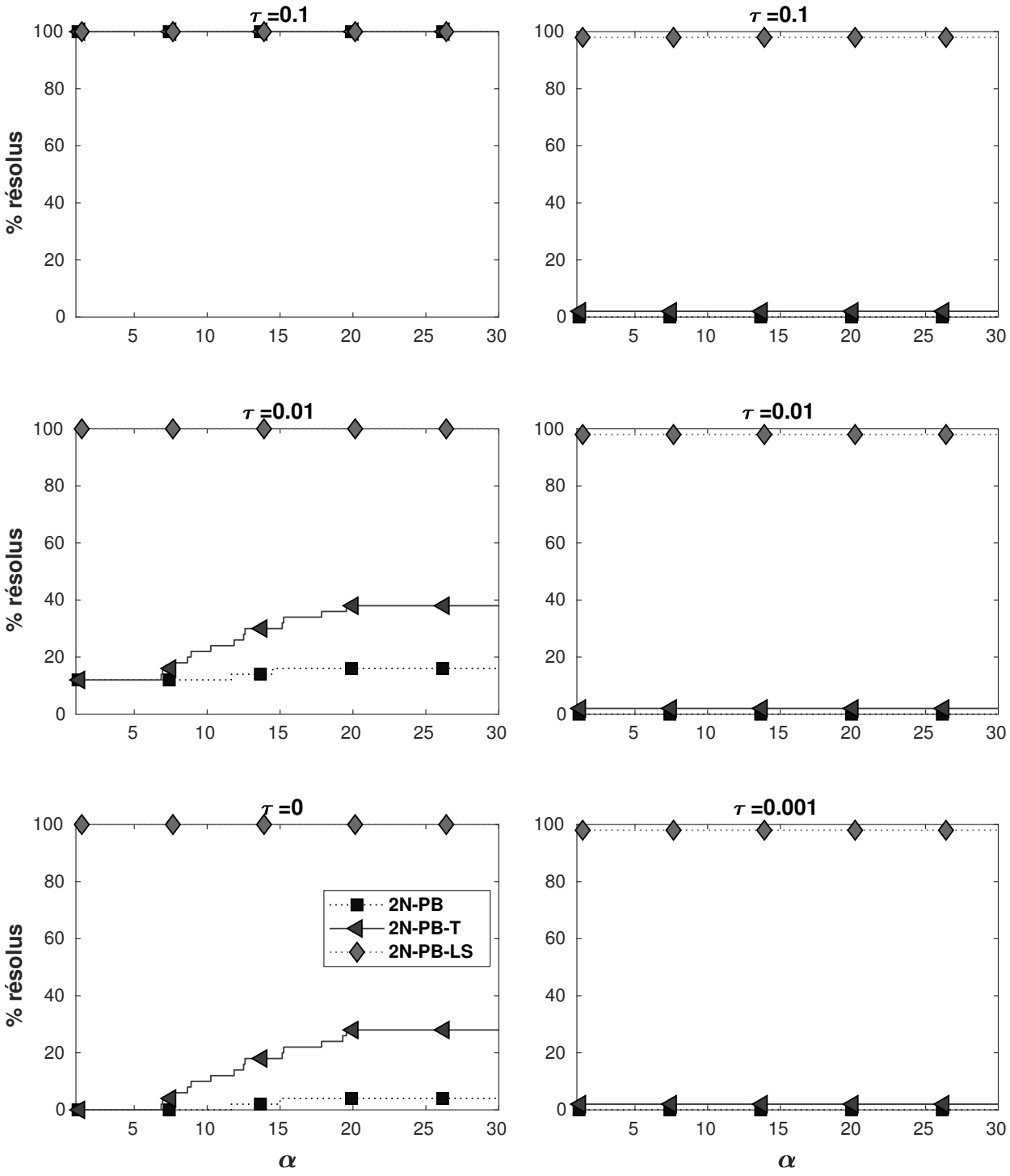
Figure 4.2 Profils de performance sur `G2` avec $n = 5$



(a) Violation des contraintes

(b) Fonction objectif

Figure 4.3 Profils de données sur `G2` avec $n = 10$



(a) Violation des contraintes

(b) Fonction objectif

Figure 4.4 Profils de performance sur `G2` avec $n = 10$

4.2.2 MDO

Ce problème d'optimisation multidisciplinaire s'attaque à la maximisation de la portée d'un avion supersonique. Une description plus détaillée de la boîte noire est donnée dans un rapport technique rédigé par Sobieszczanski-Sobieski, Agte et Sandusky [55]. MDO est composé de quatre boîtes noires distinctes où la variable de sortie de l'une peut devenir une variable d'entrée de l'autre. Parmi ces sous problèmes, on y retrouve l'aspect structurel de l'avion, l'aérodynamisme, la propulsion ainsi que la portée de l'avion. Afin d'évaluer la boîte noire MDO, dix variables d'entrées ayant un sens physique sont nécessaires telles que : l'aire de surface des ailes, le coefficient de friction de la surface avec l'air, l'altitude de vol, etc. Toutes les variables sont bornées inférieurement (l_b) et supérieurement (u_b). L'objectif principal est de maximiser la distance que l'avion peut parcourir tout en respectant un total de dix contraintes.

Construction de la matrice T

La matrice de tendance a été entièrement construite à l'aide de l'analyse d'un échantillon de cent points telle que décrite dans la section précédente. Plus particulièrement, la longueur de pas constante $\delta = (u_b - l_b)/50 \in \mathbb{R}^{10}$ a été appliquée sur les $n = 10$ éléments de cent points dans un hypercube latin. Le programme a détecté plusieurs effets monotones potentiels.

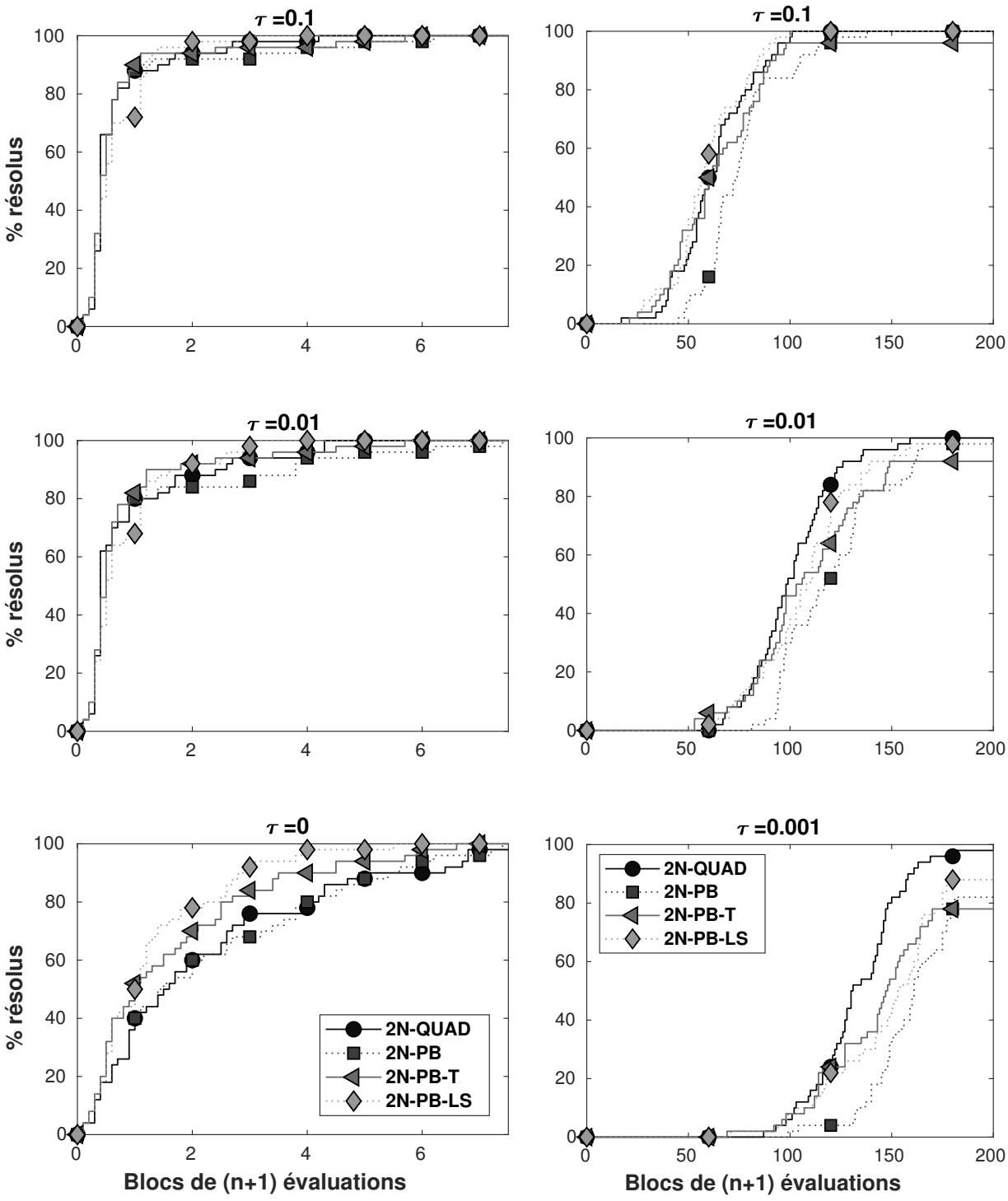
$$T = \begin{pmatrix} N/A & 1 & 1 & 1 & 1 & 1 & 0 & N/A & N/A & 0 & 0 \\ 1 & N/A & N/A & N/A & N/A & N/A & 0 & N/A & N/A & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & -1 & 1 & 0 & 0 \\ N/A & -1 & -1 & -1 & -1 & -1 & 0 & 1 & -1 & 1 & 1 \\ N/A & N/A & N/A & N/A & N/A & N/A & 1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 & 0 & 1 & -1 & 1 & -1 \\ N/A & 1 & 1 & 1 & 1 & 1 & 0 & -1 & 1 & -1 & 1 \\ 1 & N/A & N/A & N/A & N/A & N/A & 0 & N/A & N/A & 0 & 0 \\ N/A & N/A & N/A & N/A & N/A & N/A & 0 & N/A & N/A & 0 & 0 \\ N/A & N/A & N/A & 1 & 1 & 1 & 0 & -1 & 1 & N/A & N/A \end{pmatrix}_{10\times 11}$$

Bien qu'une matrice de tendance ait été construite, ce problème test garde son statut de boîte noire car, même si un sens physique pouvait être attribué aux variables et aux contraintes, le sens de la croissance n'était pas connu à priori. Le programmeur qui a construit MDO n'a pas été contacté afin de confirmer les signes retrouvés dans T. En fait, soupçonner l'existence d'une monotonie, sans connaître le sens exact de la croissance, n'est pas suffisant pour répondre à la définition 2.2 d'une boîte grise.

Résultats

Tout comme dans le problème G2 précédent, les tests numériques ont été lancés sur 50 points de départ. Ils ont été générés simplement par un hypercube latin borné par l_b et u_b. Un quatrième algorithme a aussi été testé : 2N-QUAD.

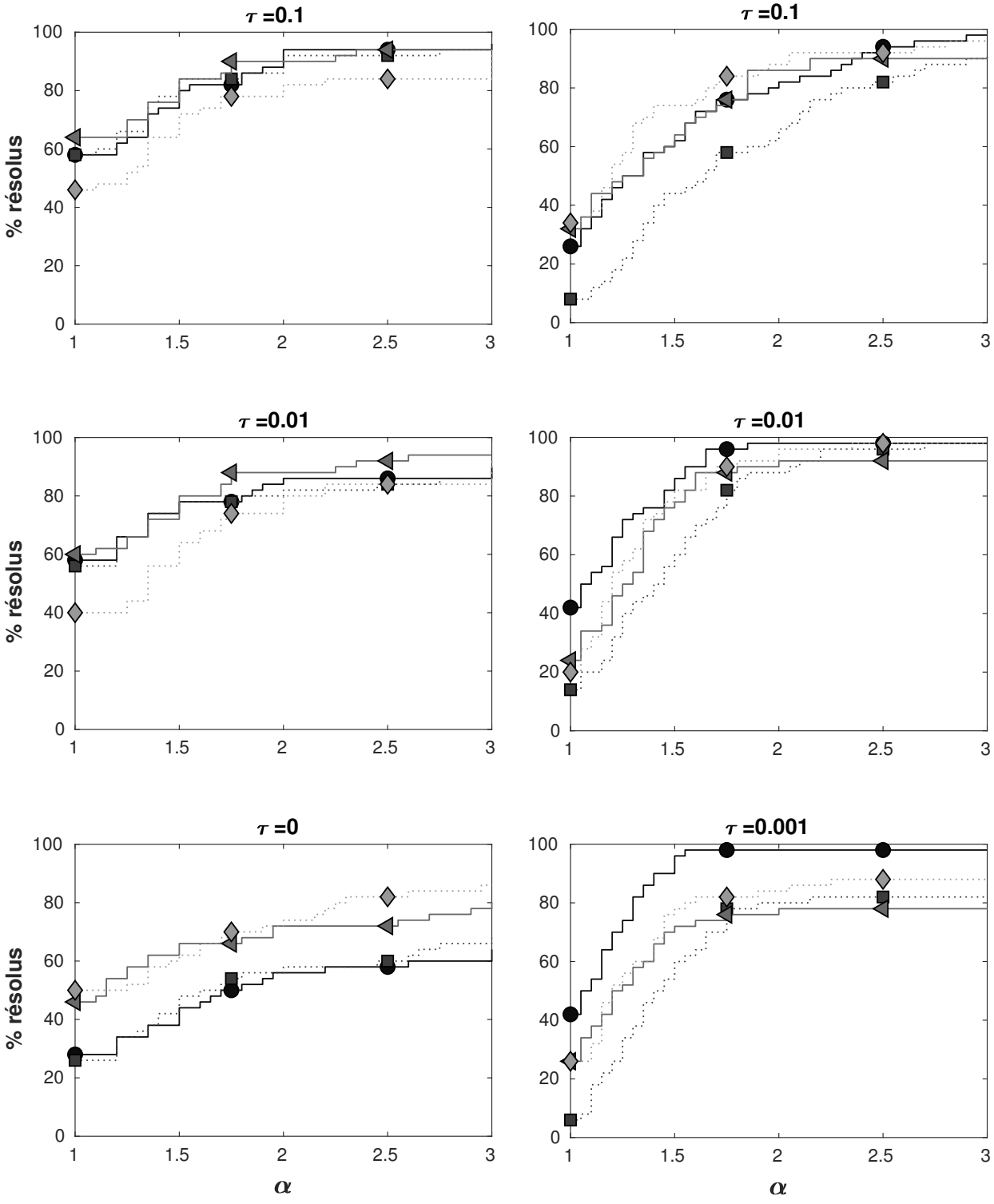
Selon les graphiques présentés dans les figures 4.5 et 4.6, les algorithmes 2N-PB-T et 2N-PB-LS dominent au niveau de l'amélioration de la fonction h représentée par les colonnes de gauche de ces deux figures. Cependant, l'utilisation des modèles quadratiques devient plus avantageux lors de la recherche d'une solution optimale pour la fonction objectif selon les figures 4.5b et 4.6b. Notons que les deux algorithmes intégrant les informations de la matrice T restent avantageux par rapport à l'algorithme de base 2N-PB.



(a) Violation des contraintes

(b) Fonction objectif

Figure 4.5 Profils de données sur `MDO`



(a) Violation des contraintes

(b) Fonction objectif

Figure 4.6 Profils de performance sur `MDO`

4.2.3 `Kemano`

Le problème `Kemano` est une boîte noire industrielle utilisée hebdomadairement par les ingénieurs de Rio Tinto afin de les guider dans la prise de décision sur la gestion de leur système hydroélectrique. Selon les prévisions météorologiques à venir, les ingénieurs déterminent s'il est plus avantageux de vider ou remplir un bassin. Le programme simule, à l'aide de scénarios d'apports en eau au réservoir, la gestion du système hydroélectrique afin d'obtenir une approximation des revenus espérés ainsi que des risques d'inondation et de pénurie d'énergie. Selon les prévisions d'apports, les ingénieurs déterminent les quantités d'eau à relâcher du réservoir afin de maximiser l'efficacité du système tout en assurant de minimiser les risques d'inondation et de pénurie futures. En tentant de trouver les paramètres de simulation optimaux pour `Kemano`, ils ont remarqué qu'il était plus rapide de trouver une solution réalisable en quelques itérations manuelles qu'en laissant NOMAD travailler. Cependant, après avoir trouvé une solution acceptable, NOMAD est efficace pour raffiner la solution. Nous resterons très large dans la description du programme `Kemano`, car il est un logiciel propriétaire de Rio Tinto. Il est à noter que chaque évaluation de la boîte noire est de 1 minute et 35 secondes sur un ordinateur personnel.

Construction de la matrice T

`Kemano` est une boîte noire composée de 5 variables qui doivent respecter les 5 contraintes qui ont les significations suivantes :

- f_1 : Débit maximal d'eau observé à Vanderhoof
- f_2 : Probabilité d'inondation à Vanderhoof
- f_3 : Probabilité d'inondation à Cheslatta
- f_4 : Probabilité d'être en pénurie d'énergie et fermer l'aluminerie
- f_5 : Probabilité d'honorer le contrat avec BC Hydro (fournir de la puissance)

Le but est de maximiser f_0 qui est la quantité d'énergie générée par les barrages. Les bornes supérieures et inférieures finies ont été posées sur chaque variables : $l_b = (0,\ 1,\ 0,\ 1,\ 1)^\top$ et $u_b = (1000,\ 3,\ 2,\ 2,\ 2)^\top$. Le rôle de chacune des variables sur les contraintes est fourni par les ingénieurs :

$$T = \begin{pmatrix} N/A & -1 & -1 & -1 & 1 & N/A \\ N/A & -1 & -1 & -1 & 1 & N/A \\ N/A & 1 & 1 & 1 & -1 & 1 \\ N/A & 1 & 1 & 1 & -1 & 1 \\ N/A & N/A & N/A & N/A & -1 & 1 \end{pmatrix}.$$

`Kemano` est bel et bien une boîte grise selon la définition 2.2. Cependant, comme la matrice T est construite à partir de l'intuition de l'utilisateur, les informations qu'elle contient peuvent être inexactes. C'est pour cette raison qu'elles ont été testées à l'aide de la méthode par échantillon utilisée sur `MDO`. Cette fois-ci, trois pas différents ont été utilisés : $\delta_5 = (u_b - l_v)/5$, $\delta_{10} = (u_b - l_v)/10$ et $\delta_{50} = (u_b - lb)/50$. De ces trois vérifications, il en résulte respectivement trois corrections de la matrice de tendance :

$$T_{C_5} = \begin{pmatrix} N/A & -1 & N/A & N/A & 0 & N/A \\ 1 & -1 & -1 & -1 & 0 & +1 \\ N/A & 1 & N/A & N/A & -1 & N/A \\ 1 & 1 & 1 & N/A & -1 & N/A \\ N/A & 1 & N/A & -1 & -1 & 1 \end{pmatrix}.$$

$$T_{C_{10}} = \begin{pmatrix} N/A & N/A & -1 & N/A & N/A & 0 \\ N/A & N/A & -1 & -1 & 1 & N/A \\ 1 & 1 & 1 & N/A & -1 & N/A \\ 1 & 1 & 1 & N/A & -1 & N/A \\ N/A & 1 & -1 & N/A & -1 & 1 \end{pmatrix}.$$

$$T_{C_{50}} = \begin{pmatrix} N/A & N/A & N/A & N/A & 0 & N/A \\ N/A & N/A & N/A & N/A & 0 & N/A \\ N/A & N/A & N/A & N/A & -1 & N/A \\ 1 & N/A & N/A & N/A & -1 & N/A \\ N/A & 1 & N/A & N/A & -1 & 1 \end{pmatrix}.$$

Les différences entre ces trois dernières matrices démontrent l'impact de la taille du pas δ choisi dans la méthode d'échantillonnage. Il devient alors difficile de choisir la matrice la plus adéquate à utiliser pour les méthodes `T` et `LS`. Cependant, deux observations importantes peuvent être soulevées. Tout d'abord, lorsqu'un élément unitaire (-1 ou 1) se retrouve à la fois dans les matrices T_{C_5} ou $T_{C_{10}}$ et dans T, le signe est conservé. Cela indique que l'utilisateur possède une bonne connaissance générale du problème et qu'il n'a pas commis d'erreur d'inattention concernant le sens de la croissance au moment d'entrer les informations dans la matrice T. Cependant l'apparition d'éléments N/A montre que son intuition générale n'est pas vraie en tout point. Cela conduit à une deuxième observation importante concernant la matrice $T_{C_{50}}$ qui contient très peu d'éléments significatifs. La perte d'information, retrouvée dans $T_{C_{50}}$, engendrée par un pas plus petit δ_{50}, indique que les contraintes sont bruitées. La figure 4.7 contient les graphes de quelques contraintes tracés en fonction d'une seule variable à partir du point de départ $x_0 = (l_b + u_b)/2$. Celle-ci démontre que les contraintes sont bel et

bien croissantes (ou décroissantes) de façon globale, mais contiennent de petits sauts ou des instabilités. Ces deux observations montrent que les informations fournies par l'utilisateur ne sont que des indications grossières du comportement de la boîte grise, mais ne permettent pas d'affirmer qu'elle est monotone au sens de la définition 3.4.

Parmi les quatre variantes de la matrice de tendance présentées, seulement deux ont été testées, soient T et T_{C_5}. Puisque les matrices T_{C_5} et $T_{C_{10}}$ représentent toutes deux une correction grossière avec un pas large et qu'elles contiennent sensiblement le même nombre d'informations utiles, une seule d'entre elles a été choisie pour mener les tests numériques. Quant à $T_{C_{50}}$, elle a été rejetée, car elle contient trop peu d'information pour avoir un impact significatif lors de l'optimisation de `Kemano`.

Résultats

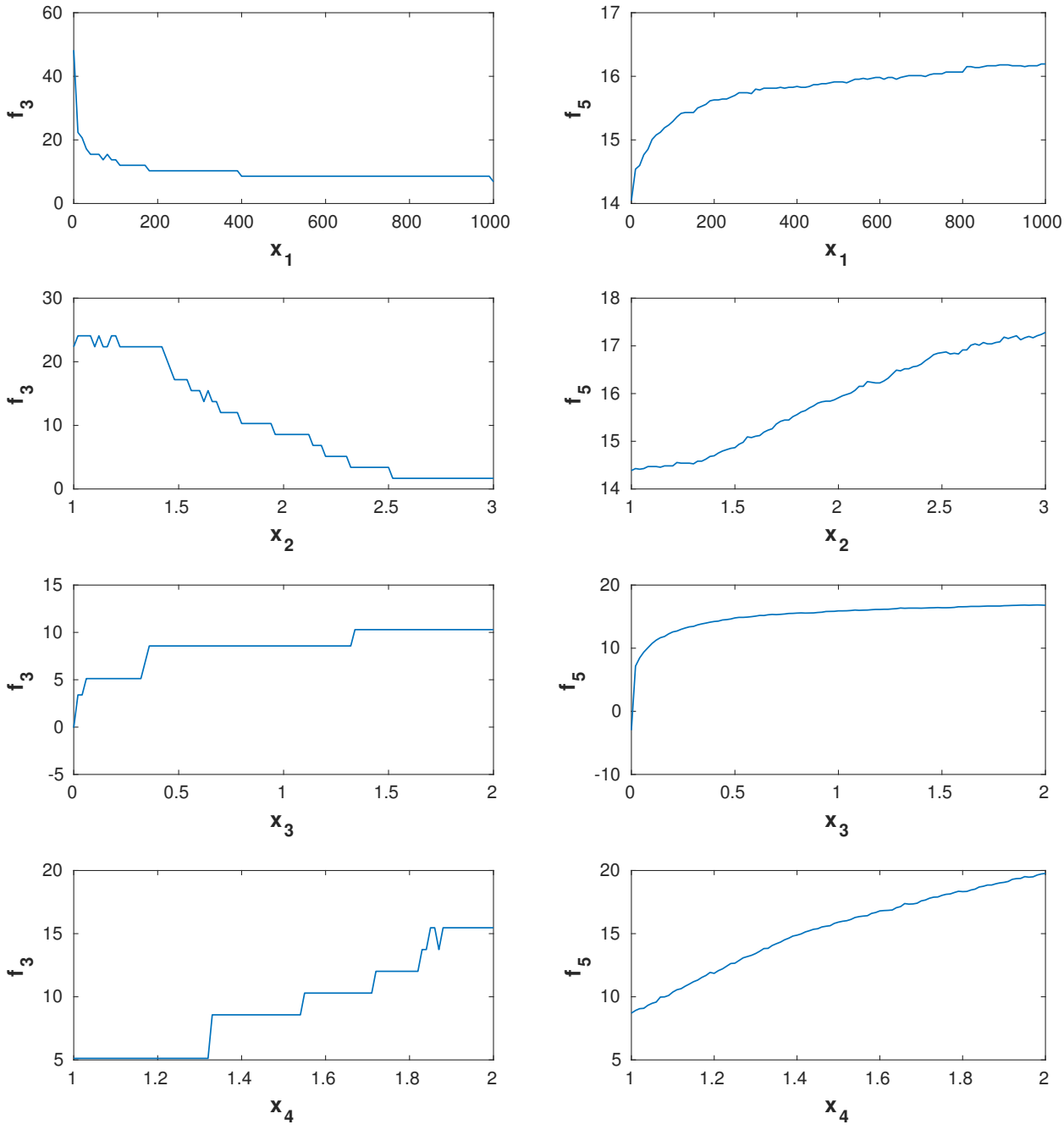
Puisque le problème `Kemano` est une boîte noire lourde à évaluer, seulement 10 points de départs ont été générés à l'aide d'un hypercube latin. Deux séries de tests différentes ont été lancées afin d'observer les effets de T et T_{C_5} qui sont respectivement présentées dans les figures 4.8 et 4.9 ainsi que 4.10 et 4.11.

Lorsque T est utilisé, les figures 4.8a et 4.9a montrent une certaine dominance de 2N-PB-T et 2N-PB-LS sur l'amélioration de h. Par contre, le comportement est similaire à celui de MADS de base (2N-PB) lorsque c'est T_{C_5} qui est utilisée (voir figures 4.10a et 4.11a).

Si, comme dans le cas présent, l'intuition de l'utilisateur n'est valable qu'au point de vue globale du problème, alors la matrice de tendance ne devrait être utile qu'au début de l'optimisation lorsque le treillis de MADS est large. Les résultats obtenus suggèrent qu'il serait alors préférable d'utiliser la matrice de tendance originale T, qui contient plus d'informations, afin d'approcher plus rapidement du domaine réalisable. En d'autres termes, même si les effets monotones indiqués par T ne sont pas nécessairement vrais en tout points, l'information grossière qu'elle contient peut-être suffisante au début de l'optimisation pour observer une amélioration sur la fonction h. Ce comportement pourrait venir pallier au manque de précision des modèles quadratiques généralement observé en début d'optimisation. Plusieurs points sont parfois nécessaires avant que les modèles quadratiques ne deviennent assez précis. Le graphique de convergence montré par la figure 4.12 représente l'amélioration normalisée de h en fonction du nombre d'évaluations de la boîte noire par les algorithmes 2N-PB-LS et 2N-QUAD. La matrice de tendance qui a servi dans la recherche linéaire est celle de l'utilisateur. Le graphique montre que 2N-QUAD stagne sur deux points de départ sur dix dans les 150 premières évaluations de la fonction, tandis que 2N-PB-LS permet à l'algorithme de bouger et d'améliorer h rapidement dans les 10 premières évaluations de la fonction sur tous

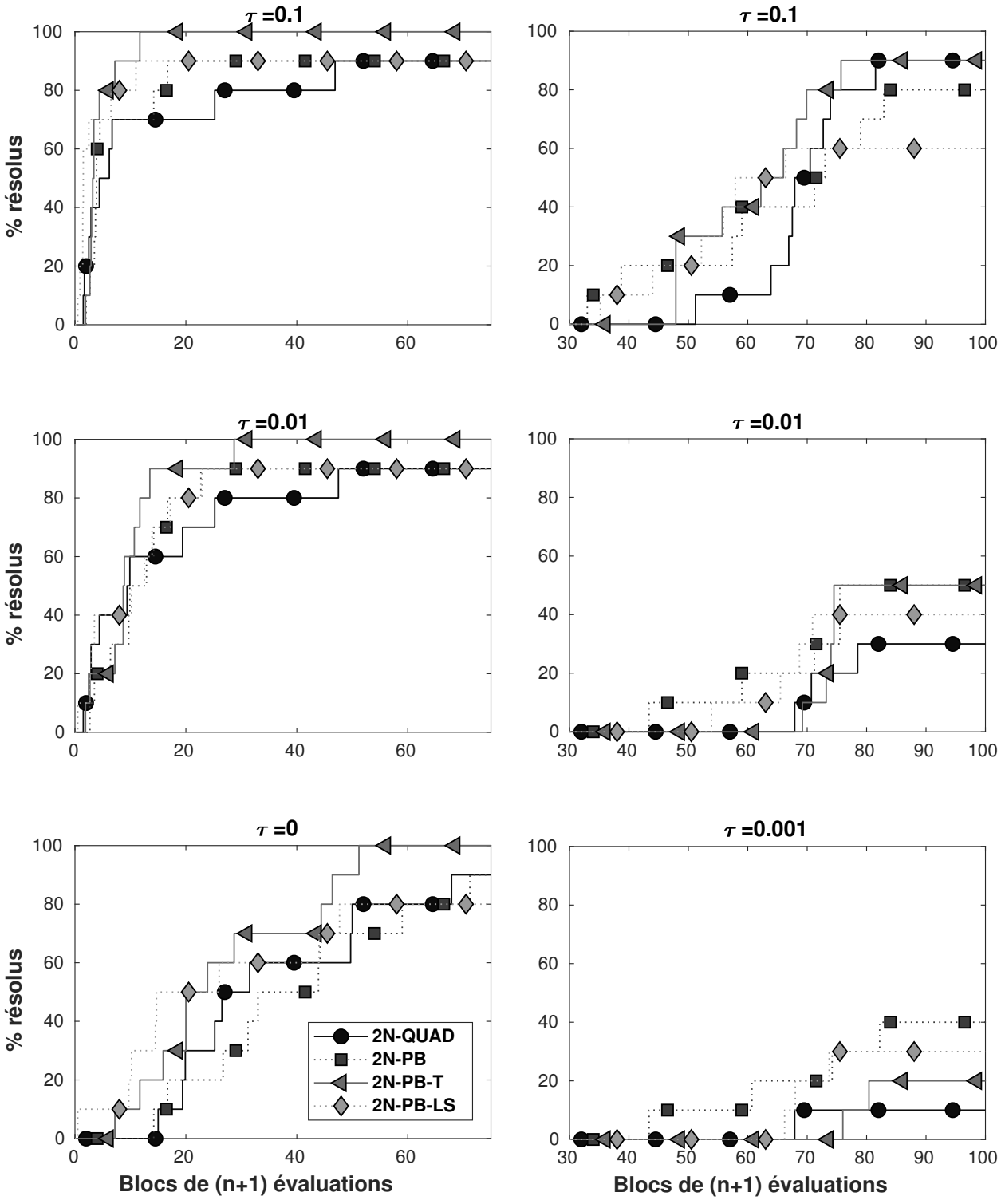
les points de départ. Sous la barre des 140 évaluations, 2N-PB-LS a trouvé une solution réalisable au problème `Kemano` dans 9 cas sur 10, tandis que 2N-QUAD n'a réussi que dans 7 cas sur 10.

Les test numériques suggèrent que la matrice de tendance T fournie par l'utilisateur permet une amélioration plus rapide de la satisfaction des contraintes, en début d'optimisation lorsque la boîte noire est bruitée, que NOMAD avec et sans modèles quadratiques ainsi que LS et T avec une matrice de tendance construite par échantillonnage.



(a) f_3 (b) f_5

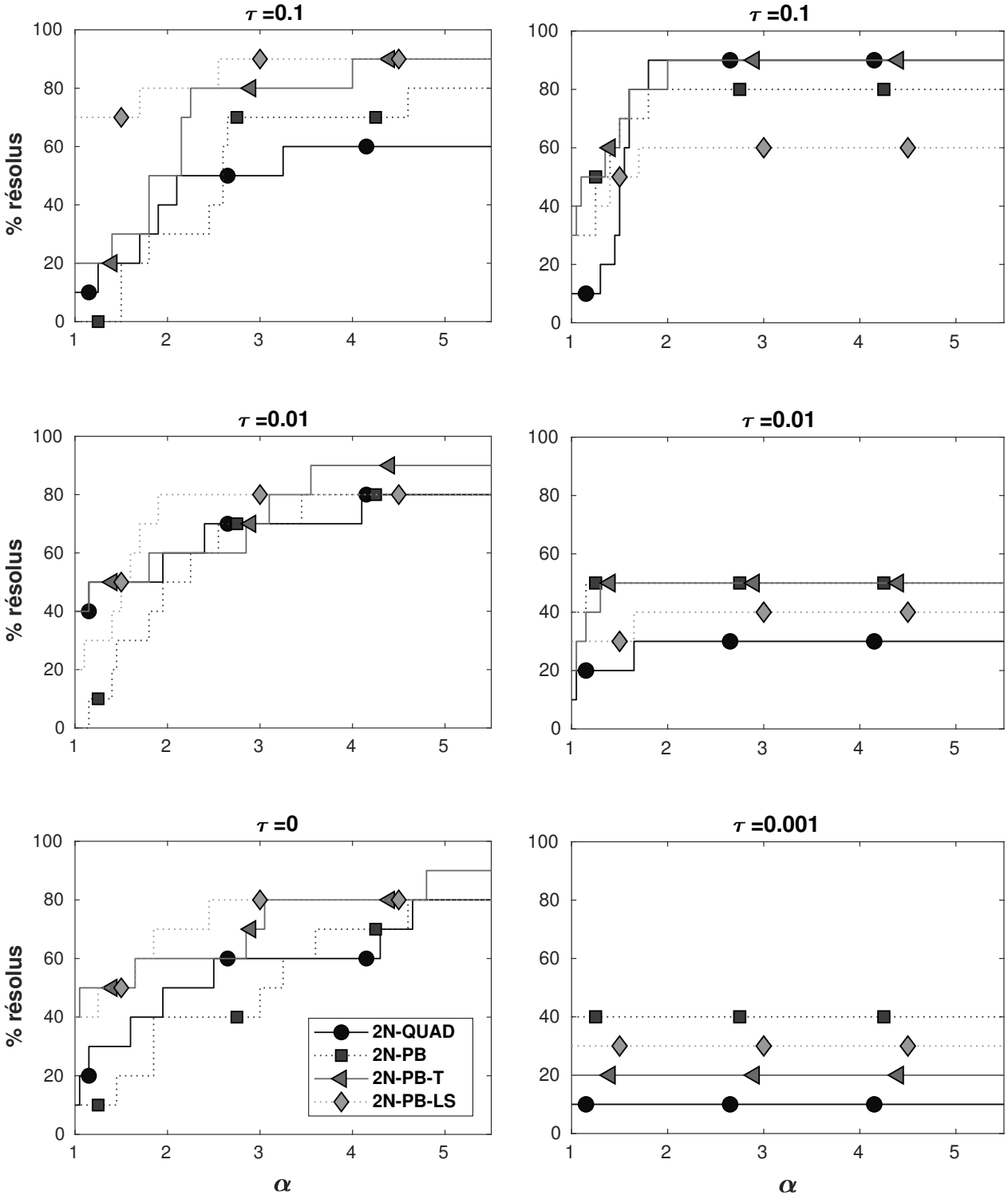
Figure 4.7 Comportement des contraintes de `Kemano`



(a) Violation des contraintes

(b) Fonction objectif

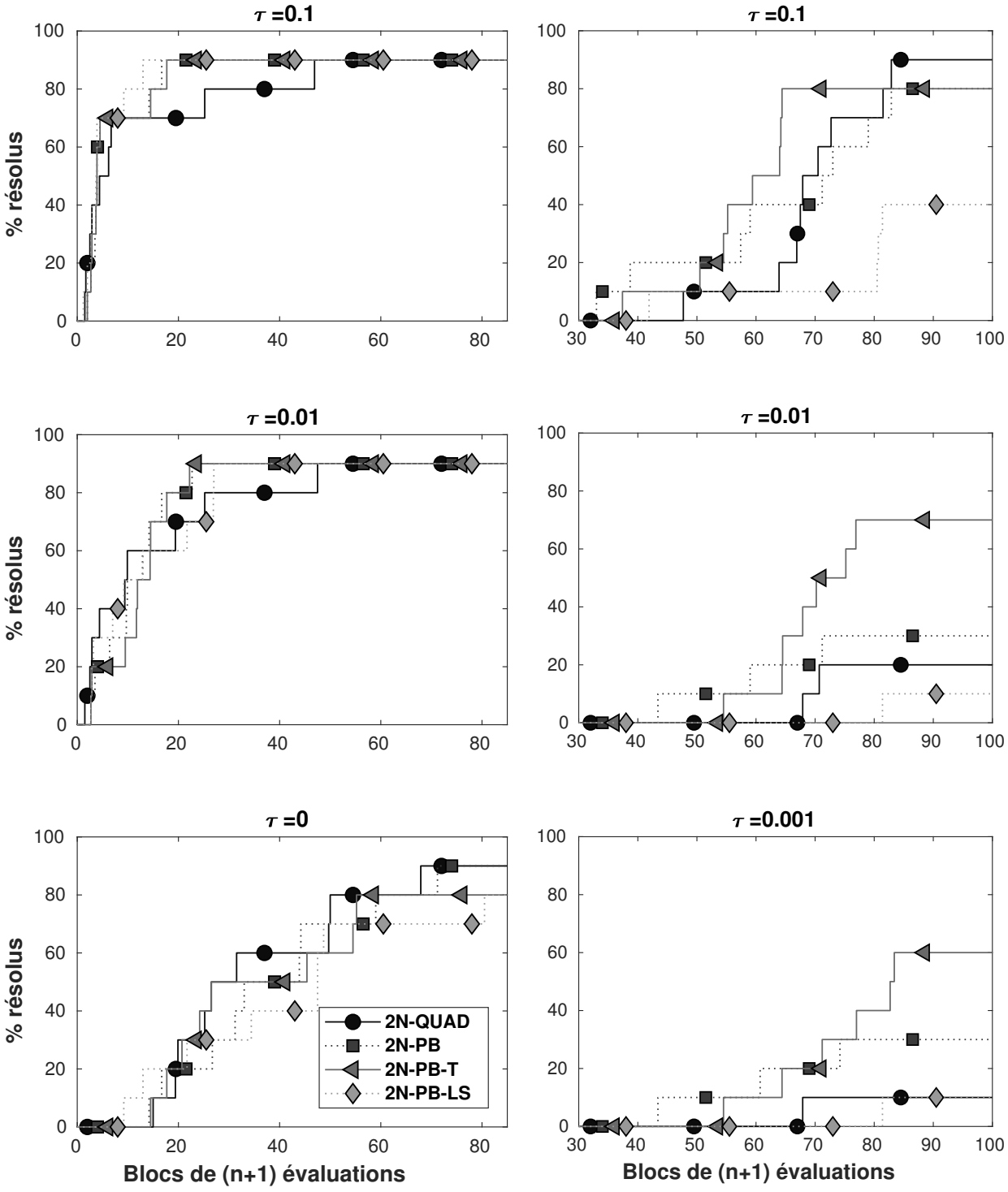
Figure 4.8 Profils de données sur `Kemano` avec T



(a) Violation des contraintes

(b) Fonction objectif

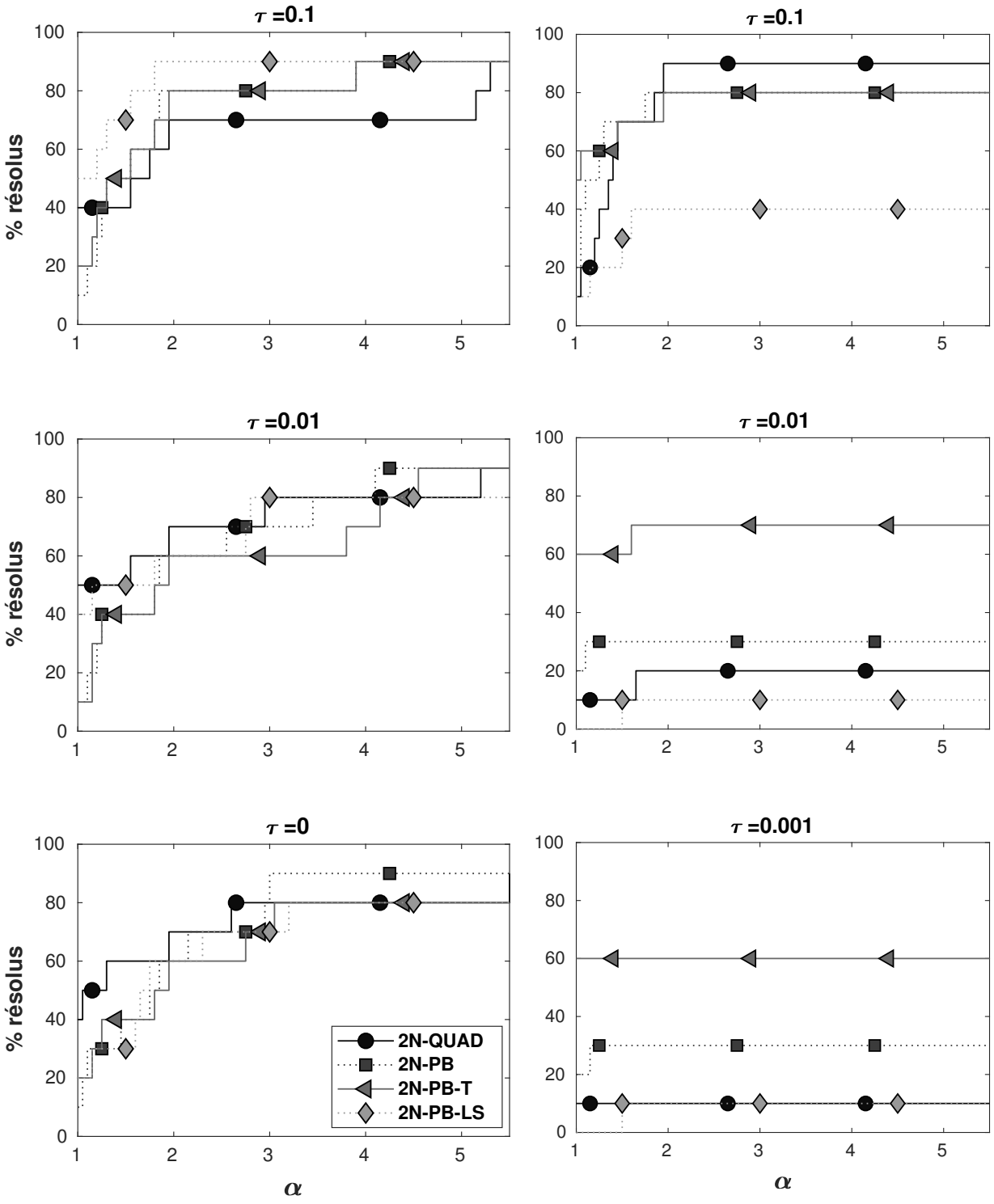
Figure 4.9 Profils de performance sur `Kemano` avec T



(a) Violation des contraintes

(b) Fonction objectif

Figure 4.10 Profils de données sur `Kemano` avec T_{C_5}



(a) Violation des contraintes

(b) Fonction objectif

Figure 4.11 Profils de performance sur `Kemano` avec T_{C_5}

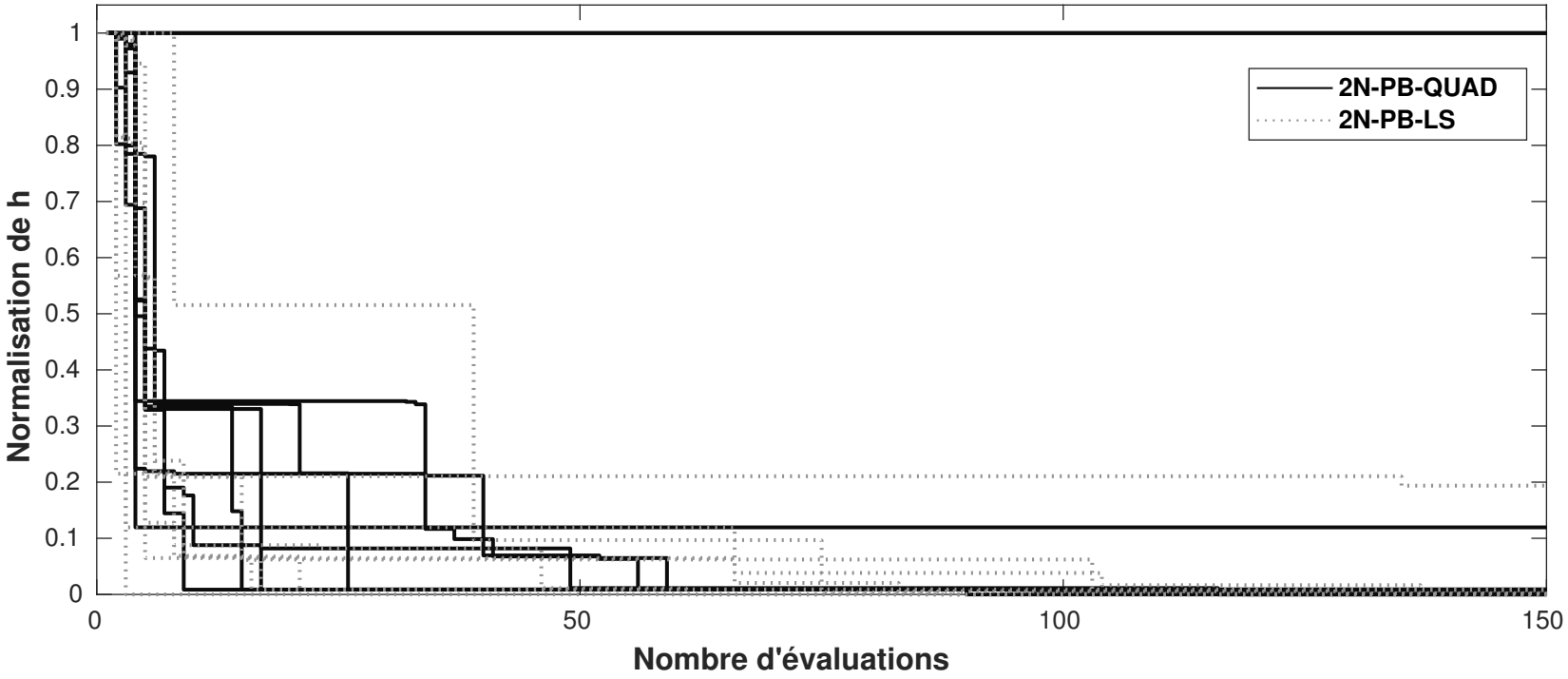


Figure 4.12 Profil de convergence sur h du problème `Kemano` avec la matrice de tendance T fournie par l'utilisateur.

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

Dans ce projet, nous avons proposé deux méthodes afin que MADS puisse exploiter des informations sur la monotonie complète ou partielle des $m+1$ fonctions définissant le problème d'optimisation. Ces deux méthodes touchent deux étapes bien différentes de l'algorithme MADS : l'ordonnancement des directions candidates dans la sonde locale T et l'ajout d'une recherche linéaire LS dans l'étape de recherche globale. Elles reposent toutes deux sur le même outil introduit à la fin du chapitre 3, soit la matrice de tendance T et la direction de tendance d_T qui en découle. Le fondement de cet outil repose sur une étude théorique de la monotonie d'une fonction à plusieurs variables sur des cônes, ce qui, au meilleur de nos connaissances, n'avait jamais été réalisé dans le cadre de la DFO.

Les méthodes ont été testées sur les problèmes G2, MDO et Kemano. T et LS ont été comparées à MADS avec et sans l'utilisation de modèles quadratiques. Dans tous les cas, le nombre de directions générées par la sonde locale a été posé à $2\times n$ et la barrière progressive a été activée. Seule l'amélioration de la fonction h, quantifiant la violation des contraintes, a été étudiée et discutée. Trois stratégies pour construire la matrice de tendance T ont été proposées : analytiquement, par échantillonnage et selon l'intuition de l'utilisateur. Les problèmes tests présentés appellent chacune une méthode de construction différente de T.

5.2 Discussion générale et travaux futurs

Le travail fut principalement axé sur l'optimisation en deux phases, soit l'amélioration de la violation des contraintes h et la minimisation de l'objectif une fois qu'un point réalisable a été trouvé. C'est pour cette raison que seuls des points de départs non réalisables ont été générés sur les problèmes test. La première phase a permis d'observer l'impact direct de l'utilisation de T lors d'une optimisation. Il aurait été plus difficile de tirer la même qualité d'information à partir de points réalisables, car NOMAD utilise la barrière progressive qui permet d'optimiser autour de deux itérés courants. L'appel à l'ordonnancement avec T aurait alors été inégal dans le cas ou le statut de l'itéré courant réalisable aurait passé de primaire à secondaire par exemple. Parfois, il n'y aurait eu que deux directions à ordonnancer, ce qui n'est pas significatif pour un problème dans $\mathbb{R}^5$. Aussi, lorsque l'optimum se trouve sur la bordure du domaine réalisable $\partial\Omega$, ce qui est nécessairement le cas pour une fonction K-monotone sur un compact non vide avec $K\backslash\{0\} \neq \emptyset$ (voir proposition 3.1), la minimisation de f_0 requiert

de sortir de Ω alors que le respect des contraintes demande le contraire. Cette contradiction peut se refléter dans la matrice T et conséquemment créer une direction de tendance d_T nulle, ce qui rend LS et T complètement inutile. Toutes ces raisons nous laissent croire que l'usage de T est efficace pour l'amélioration des contraintes, mais pas nécessairement pour f_0.

Selon la nature du problème test, différentes méthodes pour construire la matrice T semblent être plus avantageuses. Dans le cas de G2, les informations de T ont été trouvées directement grâce aux équations analytiques connues. La recherche linéaire LS s'est grandement démarquée des autres algorithmes, plus particulièrement lorsque la dimension du problème s'est vu augmentée. La construction de T par échantillonnage a obtenu une bonne performance sur MDO avec l'algorithme LS, mais pas sur Kemano où les informations approximatives fournies par l'utilisateur semblaient être plus appropriées. En fait, si le problème est lisse par morceaux (MDO), les informations sur la monotonie contenue dans T sont valables tout au long de l'optimisation. Ce n'est pas le cas pour une boîte noire bruitée (Kemano) où l'intuition de l'utilisateur n'est valable que lorsque des gros pas sont faits. Dans tous les cas, une piste à suivre serait d'utiliser T en début d'algorithme pour ensuite laisser place à une autre méthode lorsque le treillis de MADS devient trop fin (Kemano) pour le niveau de bruit de la boîte grise ou lorsque les modèles sont jugés assez précis (MDO). La recherche du moment idéal pour passer d'une méthode à l'autre pourra faire l'objet de travaux futurs.

Le problème Kemano démontre que la technique d'échantillonnage pour remplir la matrice de tendance doit être améliorée, car les matrices $T_{C_{10}}$ et T_{C_5} contiennent des éléments très différents selon la taille du pas δ choisie. Tel qu'il a été souligné à la section 4.1.4, un pas δ aléatoire pourrait répondre à ce problème. Une seconde idée serait d'utiliser une matrice de tendance qui s'ajuste selon l'échelle du treillis de MADS en cours d'optimisation.

Dans ce travail, l'optimisation de boîtes noires à l'aide de modèles substituts n'a pas été abordée. Cependant, l'exploitation de structures monotones pourraient y être intégré. En 2016, Audet, Kokkolaras, Le Digabel et Talgorn ont proposé une technique de gestion d'une banque de modèles substituts afin de les intégrer dans l'étape de recherche globale de MADS [13]. L'idée est de cerner les modèles les plus adéquats à l'aide d'une métrique sur la conservation de l'ordre des points. En d'autres termes, un modèle $\hat{f}_0$ est jugé bon pour f_0 si $\hat{f}_0(x) \leq \hat{f}_0(y) \Leftrightarrow f_0(x) \leq f_0(y)$. La condition sur les contraintes diffère légèrement : $\hat{f}_j$ est jugé bon pour f_j si $\hat{f}_j \leq 0 \Leftrightarrow f_j \leq 0$. Cependant, il y a un «trou» dans la métrique lorsqu'aucune solution réalisable n'est connue. En fait, la matrice de tendance s'intègre bien dans cette idée de conservation de l'ordre entre les points grâce au concept de la monotonie et pourrait pallier à ce trou. L'idée est bien simple : lancer la méthode d'échantillonnage introduite dans la section 4.1.4 sur la collection des modèles substituts $\hat{f}_j$ afin de cibler ceux

qui respectent le mieux les signes retrouvés dans la matrice de tendance T. La technique serait peu coûteuse en temps, puisque aucune évaluation de la boîte noire n'est demandée.

Finalement, les méthodes présentées dans ce travail laissent beaucoup de liberté dans leur implémentation. Nous n'avons qu'à penser à la recherche linéaire LS qui peut prendre différents points ou à l'ordonnancement T qui peut être jumelée à une autre méthode afin de créer un système de pointage plus complexe pour noter la priorité de chaque direction candidate. De plus, le calcul de d_T peut inclure les informations connues sur la fonction objectif en ajoutant 0 dans l'ensemble $J(x^k)$, et ce, même si x^k est non réalisable. Il y a donc place à imagination et à la création de plusieurs autres variantes.

RÉFÉRENCES

[1] M. Abramson, C. Audet, G. Couture, J. E. Dennis, Jr., S. Le Digabel, C. Tribes et V. R. Montplaisir, *The NOMAD project*. Software available at `https://www.gerad.ca/nomad/`.

[2] M. Abramson, C. Audet et J. E. Dennis, Jr., *Generalized pattern searches with derivative information*, Mathematical Programming, Series B, 100 (2004), p. 3–25.

[3] L. Adjengue, C. Audet et I. B. Yahia, *A variance-based method to rank input variables of the Mesh Adaptive Direct Search algorithm*, Optimization Letters, 8 (2014), p. 1599–1610.

[4] N. Amaioua, C. Audet, A. R. Conn et S. L. Digabel, *Efficient solution of quadratically constrained quadratic subproblems within the mesh adaptive direct search algorithm*, European Journal of Operational Research, 268 (2018), p. 13 – 24.

[5] C. Audet, *Convergence Results for Generalized Pattern Search Algorithms are Tight*, Optimization and Engineering, 5 (2004), p. 101–122.

[6] C. Audet, *A Survey on Direct Search Methods for Blackbox Optimization and Their Applications*, Springer New York, 2014, p. 31–56.

[7] C. Audet et J. E. Dennis, Jr., *A pattern search filter method for nonlinear programming without derivatives*, SIAM Journal on Optimization, 14 (2004), p. 980–1010.

[8] C. Audet et J. E. Dennis, Jr., *Mesh adaptive direct search algorithms for constrained optimization*, SIAM Journal on Optimization, 17 (2006), p. 188–217.

[9] C. Audet et J. E. Dennis, Jr., *A progressive barrier for derivative-free nonlinear programming*, SIAM Journal on Optimization, 20 (2009), p. 445–472.

[10] C. Audet et W. Hare, *Derivative-Free and Blackbox Optimization*, Springer Series in Operations Research and Financial Engineering, Springer International Publishing, 2017.

[11] C. Audet, A. Ianni, S. Le Digabel et C. Tribes, *Reducing the Number of Function Evaluations in Mesh Adaptive Direct Search Algorithms*, SIAM Journal on Optimization, 24 (2014), p. 621–642.

[12] C. Audet, J. J. E. Dennis et S. L. Digabel, *Parallel space decomposition of the mesh adaptive direct search algorithm*, SIAM Journal on Optimization, 19 (2008), p. 1150–1170.

[13] C. Audet, M. Kokkolaras, S. L. Digabel et B. Talgorn, *Order-based error for managing ensembles of surrogates in mesh adaptive direct search*, Journal of Global Optimization, 70 (2018), p. 645–675.

[14] C. Audet, S. Le Digabel et C. Tribes, *Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization*, Optimization and Engineering, 17 (2016), p. 333–358.

[15] C. Audet et C. Tribes, *Mesh-based nelder-mead algorithm for inequality constrained optimization*, Les Cahiers du GERAD G-2017-90, GERAD, Rap. tech., Canada, 2017.

[16] I. Bajaj, S. Iyer et M. M. F. Hasan, *A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point*, Computers and Chemical Engineering, (2017).

[17] T. Bohlin, *Practical Grey-box Process Identification : Theory and Applications*, Springer Verlag London Limited, 1. aufl. éd., 2006.

[18] J. M. Borwein, J. V. Burke et A. S. Lewis, *Differentiability of cone-monotone functions on separable banach space*, Proceedings of the American Mathematical Society, 132 (2004), p. 1067–1076.

[19] F. Boukouvala et C. A. Floudas, *Argonaut : Algorithms for global optimization of constrained grey-box computational problems*, Optimization Letters, 11 (2017), p. 895–913.

[20] F. Boukouvala, M. M. F. Hasan et C. A. Floudas, *Global optimization of general constrained grey-box models : new method and its application to constrained pdes for pressure swing adsorption*, Journal of Global Optimization, 67 (2017), p. 3–42.

[21] G. Box, *Evolutionary operation : A method for increasing industrial productivity*, Appl. Statist., 6 (1957), p. 81–101.

[22] K. Chetehouna, O. Sero-Guillaume, I. Sochet et A. Degiovanni, *On the experimental determination of flame front positions and of propagation parameters for a fire*, International Journal Of Thermal Sciences, 47 (2008), p. 1148–1157.

[23] F. H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[24] A. R. Conn, N. I. M. Gould et P. L. Toint, *Trust-region methods : Andrew R. Conn, Nicholas I.M. Gould, Philippe L. Toint*, vol. 1, Society for Industrial and Applied Mathematics, Philadelphia, Pa, 2000.

[25] A. R. Conn et S. Le Digabel, *Use of quadratic models with mesh-adaptive direct search for constrained black box optimization*, Optimization Methods and Software, 28 (2013), p. 139–158.

[26] A. R. Conn, K. Scheinberg et L. Vicente, *Introduction to Derivative-Free Optimization*, MOS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[27] P. Côté, *Communication privée*, 2016-09-01.

[28] C. Davis, *Theory of positive linear dependence*, American Journal of Mathematics, 76 (1954), p. 733–746.

[29] M. Delfour, *Introduction à l'optimisation et au calcul semi-différentiel*, DUNOD, 2012.

[30] E. Dolan et J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), p. 201–213.

[31] J. E. Dennis, Jr. et V. Torczon, *Direct search methods on parallel machines*, SIAM Journal on Optimization, 1 (1991), p. 448–474.

[32] J. P. Eason et L. T. Biegler, *A trust region filter method for glass box/black box optimization*, AIChE Journal, 62 (2016), p. 3124–3136.

[33] E. Fermi et N. Metropolis, *Numerical solution of a minimum problem*, Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA, 1952.

[34] R. Fletcher et S. Leyffer, *Nonlinear programming without a penalty function*, Mathematical Programming, 91 (2002), p. 239–269.

[35] N. Gould et J. Scott, *A note on performance profiles for benchmarking software*, ACM Transactions on Mathematical Software, 43 (2016), p. 1–5.

[36] I. Griva, S. G. Nash et A. Sofer, *Linear and nonlinear optimization*, vol. 108, Siam, 2009.

[37] N. Hansen, *The CMA Evolution Strategy : A Comparing Review*, *in* Towards a New Evolutionary Computation, J. Lozano, P. Larrañaga, I. Inza et E. Bengoetxea, éds., vol. 192 de Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg, 2006, p. 75–102.

[38] W. Hock et K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, vol. 187 de Lecture Notes in Economics and Mathematical Systems, Springer, Berlin, Germany, 1981.

[39] R. Hooke et T. Jeeves, *"Direct Search" Solution of Numerical and Statistical Problems*, Journal of the Association for Computing Machinery, 8 (1961), p. 212–229.

[40] S. Le Digabel, *Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS algorithm*, ACM Transactions on Mathematical Software, 37 (2011), p. 44 :1–44 :15.

[41] S. Le Digabel et S. M. Wild, *A taxonomy of constraints in simulation-based optimization*, Rap. tech. G–2015–57, Les cahiers du GERAD, 2015.

[42] H. Lebesgue, *Sur le probleme de Dirichlet*, Tip. Matematca, 1907.

[43] R. Lewis et V. Torczon, *Pattern search methods for linearly constrained minimization*, SIAM Journal on Optimization, 10 (2000), p. 917–941.

[44] G. Liuzzi et A. Risi, *A decomposition algorithm for unconstrained optimization problems with partial derivative information*, Optimization Letters, 6 (2012), p. 437–450.

[45] M. McKay, R. Beckman et W. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 21 (1979), p. 239–245.

[46] Z. Michalewicz et M. Schoenauer, *Evolutionary algorithms for constrained parameter optimization problems*, Evolutionary computation, 4 (1996), p. 1–32.

[47] J. Moré et S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM Journal on Optimization, 20 (2009), p. 172–191.

[48] G. D. Mostow, *Quasi-conformal mappings inn-space and the rigidity of hyperbolic space forms*, Publications mathématiques de l'IHÉS, 34 (1968), p. 53–104.

[49] J. Nelder et R. Mead, *A simplex method for function minimization*, The Computer Journal, 7 (1965), p. 308–313.

[50] M. Prechtel, G. Leugering, P. Steinmann et M. Stingl, *Towards optimization of crack resistance of composite materials by adjustment of fiber shapes*, Engineering Fracture Mechanics, 78 (2011), p. 944–960.

[51] R. G. Regis, *On the properties of positive spanning sets and positive bases*, Optimization and Engineering, 17 (2016), p. 229–262.

[52] L. Rios et N. Sahinidis, *Derivative-free optimization : a review of algorithms and comparison of software implementations*, Journal of Global Optimization, 56 (2012), p. 1247–1293.

[53] R. Rockafellar, *Generalized directional derivatives and subgradients of nonconvex functions*, Canad. J. Math., 32 (1980), p. 257–280.

[54] A. Rubinov, H. Tuy et H. Mays, *An algorithm for monotonic global optimization problems*, Optimization, 49 (2001), p. 205–221.

[55] J. Sobieszczanski-Sobieski, J. Agte et R. Sandusky, Jr., *Bi-Level Integrated System Synthesis (BLISS)*, Rap. tech. NASA/TM-1998-208715, NASA, Langley Research Center, 1998.

[56] V. Torczon, *On the convergence of pattern search algorithms*, SIAM Journal on Optimization, 7 (1997), p. 1–25.

[57] H. Tuy, *Monotonic optimization : Problems and solution approaches*, SIAM Journal on Optimization, 11 (2000), p. 464–494.

[58] H. A. Van Dyke, K. R. Vixie et T. J. Asaki, *Cone monotonicity : Structure theorem, properties, and comparisons to other notions of monotonicity*, Abstract and Applied Analysis, 2013 (2013), p. 1–8.

[59] S. Vodopyanov et V. Goldstein, *Quasiconformal mappings and spaces of functions with generalized first derivatives*, 17 (1976), p. 399–411.

[60] B. Whiten, *Model completion and validation using inversion of grey box models*, ANZIAM Journal, 54 (2013), p 187.